(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)公開番号
特開2024-76005
(P2024-76005A)
(43)公開日　令和6年6月5日(2024.6.5)

(51)国際特許分類		FI		テーマコード（参考）
Ｇ０６Ｆ	*8/34 (2018.01)*	Ｇ０６Ｆ	8/34	５Ｂ３７６
Ｇ０６Ｆ	*3/0481(2022.01)*	Ｇ０６Ｆ	3/0481	５Ｅ５５５

審査請求　未請求　請求項の数　16　　ＯＬ　（全90頁）

(21)出願番号　特願2022-187335(P2022-187335)
(22)出願日　令和4年11月24日(2022.11.24)

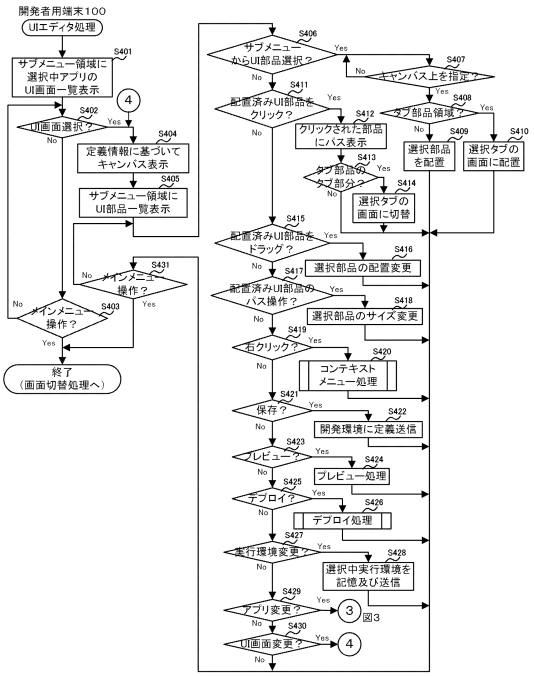
（特許庁注：以下のものは登録商標）
１．ＪＡＶＡＳＣＲＩＰＴ

(71)出願人　390002761
キヤノンマーケティングジャパン株式会社
東京都港区港南２丁目１６番６号
(71)出願人　592135203
キヤノンＩＴソリューションズ株式会社
東京都港区港南２丁目１６番６号
(74)代理人　100189751
弁理士　木村 友輔
(74)代理人　100227857
弁理士　中山 圭
(72)発明者　柴本 文洋
東京都港区港南２丁目１６番６号 キヤノンＩＴソリューションズ株式会社内
(72)発明者　清野 光広

最終頁に続く

(54)【発明の名称】　情報処理システム、情報処理システムの制御方法、及びプログラム

(57)【要約】
【課題】配置された画面部品による切替対象となる要素画面に対して、より容易に他の画面部品を配置可能な仕組みを提供する。
【解決手段】　アプリケーションの開発画面に配置された第１の種別のコンポーネントのうち指示領域に対する操作があったことに応じて、前記第１の種別のコンポーネントの領域内の少なくとも一部の領域の表示を、操作された指示領域に対応する要素画面に切り替えるように制御する表示制御手段と、
　前記開発画面に配置された前記第１の種別のコンポーネントの領域に対して他のコンポーネントを配置する操作があった場合に、前記第１の種別のコンポーネントの複数の要素画面のうち、現在表示されている要素画面に対して前記他のコンポーネントを配置するように制御する制御手段と
　を有する。
【選択図】図４



【特許請求の範囲】
【請求項１】
アプリケーションの開発画面に配置された第１の種別のコンポーネントのうち指示領域に対する操作があったことに応じて、前記第１の種別のコンポ―ントの領域内の少なくとも一部の領域の表示を、操作された指示領域に対応する要素画面に切り替えるように制御する表示制御手段と、
前記開発画面に配置された前記第１の種別のコンポーネントの領域に対して他のコンポーネントを配置する操作があった場合に、前記第１の種別のコンポーネントの複数の要素画面のうち、現在表示されている要素画面に対して前記他のコンポーネントを配置するように制御する制御手段と
を有することを特徴とする情報処理システム。
【請求項２】
前記制御手段は、前記現在表示されている要素画面の識別情報に対応づけて、当該要素画面における前記他のコンポーネントの位置を記録するように制御することを特徴とする請求項１に記載の情報処理システム。
【請求項３】
前記制御手段は、前記現在表示されている要素画面の識別情報と、当該要素画面における前記他のコンポーネントの位置とに、さらに前記第１の種別のコンポーネントの識別情報を対応付けて記憶するように制御することを特徴とする請求項２に記載の情報処理システム。
【請求項４】
前記制御手段は、前記第１の種別のコンポーネントの識別情報と、前記現在表示されている要素画面の識別情報と、当該要素画面における前記他のコンポーネントの位置とに、更に前記第１の種別のコンポーネントが配置された画面における前記第１の種別のコンポーネントの位置を記録するように制御することを特徴とする請求項３に記載の情報処理システム。
【請求項５】
前記表示制御手段は、前記開発画面において、前記第１の種別のコンポーネントの第１の要素画面と第２の要素画面とのうち、前記第１の要素画面が表示されている際に前記第２の要素画面に対応する指示領域に対する操作があったことに応じて、前記第１の要素画面に配置されていたコンポ―ネントを非表示として、前記第２の要素画面に配置されたコンポ―ントを表示するように制御することを特徴とする請求項１に記載の情報処理システム。
【請求項６】
前記表示制御手段は、前記開発画面に配置された第２の種別のコンポーネントに対する操作があっても当該第２の種別のコンポーネントの領域内の表示内容を切り替えないことを特徴とする請求項１に記載の情報処理システム。
【請求項７】
前記第２の種別のコンポ―ネントはボタンであることを特徴とする請求項６に記載の情報処理システム。
【請求項８】
前記開発画面に配置されたコンポーネントに対し、構築後の前記アプリケーションの画面において当該コンポーネントが操作された場合に実行すべき処理であるアクションを設定する設定手段をさらに有し、
前記制御手段は、前記開発画面に配置されたコンポーネントが当該開発画面において操作されても、当該コンポ―ネントに設定されたアクションを実行しないように制御することを特徴とする請求項１に記載の情報処理システム。
【請求項９】
前記表示制御手段は、前記開発画面に配置された前記第１の種別のコンポーネントの前記指示領域に対する操作があったことに応じて、操作された指示領域に対応する要素画面

に切り替えるとともに、当該第１の種別のコンポ―ントが選択されたことを示す表示物を表示するように制御することを特徴とする請求項１に記載の情報処理システム。
【請求項１０】
前記表示物に対する操作に応じて、前記第１の種別のコンポーネントの全体の表示位置と全体の表示サイズの少なくとも一方を変更することを特徴とする請求項９に記載の情報処理システム。
【請求項１１】
前記第１の種別のコンポーネントに含まれる複数の指示領域の数と順序を、当該第１の種別のコンポーネントに関する設定を行う設定画面に対する操作に応じて設定するように制御する設定手段をさらに有することを特徴とする請求項１に記載の情報処理システム。
【請求項１２】
前記第１の種別のコンポーネントは、前記指示領域としてタブを有するコンポーネントであることを特徴とする請求項１に記載の情報処理システム。
ように制御する表示制御手段と、
【請求項１３】
前記第１の種別のコンポーネントは、前記指示領域としてドロワーメニューを表示させる指示を受け付ける領域を有するコンポ―ネント、前記指示領域としてポップアップメニューを表示させる指示を受け付ける領域を有するコンポーネントのうち少なくとも１つであることを特徴とする請求項１に記載の情報処理システム。
【請求項１４】
前記第１の種別のコンポーネントの複数の要素画面のうち、前記現在表示されている要素画面に対して前記他のコンポーネントが配置されることを示す情報を含む前記アプリケーションの定義情報を、開発環境から実行環境にデプロイすることによって前記アプリケーションを構築する構築手段をさらに有することを特徴とする請求項１に記載の情報処理システム。
【請求項１５】
アプリケーションの開発画面に配置された第１の種別のコンポーネントのうち指示領域に対する操作があったことに応じて、前記第１の種別のコンポ―ントの領域内の少なくとも一部の領域の表示を、操作された指示領域に対応する要素画面に切り替えるように制御する表示制御ステップと、
前記開発画面に配置された前記第１の種別のコンポーネントの領域に対して他のコンポーネントを配置する操作があった場合に、前記第１の種別のコンポーネントの複数の要素画面のうち、現在表示されている要素画面に対して前記他のコンポーネントを配置するように制御する制御ステップと
を有することを特徴とする情報処理システムの制御方法。
【請求項１６】
少なくとも１つのコンピュータを、請求項１乃至１４のいずれか１項に記載された情報処理システムの各手段として機能させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
情報処理システム、情報処理システムの制御方法、及びプログラム関し、特に、アプリケーションソフトウェアの構築のために用いて好適な技術に関する。
【背景技術】
【０００２】
従来、プログラミング言語によるコード記述が不要または少なくてよいノーコード開発ツール／ローコード開発ツールとして、定義に従ってアプリケーションソフトウェア（以下、アプリケーションと称する）を構築するアプリケーション構築ツールが存在する。
【０００３】
特許文献１には、構築するアプリケーションの画面を編集するためのレイアウトエディ

タ画面に、ボタン、テキストボックス等の画面部品のいずれかをドラッグ＆ドロップで配置でき、配置した画面部品を入出力定義テーブルに記憶することが提案されている。
特許文献２には、GUI生成のための基本パーツとして予め持っているＧＵＩ部品（画面構成要素部品）にタブ類を含むことが開示されている。
【先行技術文献】
【特許文献】
【０００４】
【特許文献１】特開２０１９－４９８５８号公報
【特許文献２】特開２００８－３６４９号公報
【発明の概要】
【発明が解決しようとする課題】
【０００５】
構築するアプリケーションの画面を編集するためのエディタ画面（アプリケーションの開発画面）においては、配置された画面部品（ＧＵＩ部品、コンポーネント）として例えばボタンが操作（例えばクリックやタッチ）された場合、操作されたボタンを選択状態としてサイズ変更や配置位置の変更の指示などを受け付けるのが好ましい。このようにした場合、ボタン自体の機能、すなわち構築されたアプリケーションで当該ボタンが操作された場合の機能はエディタ画面では実行されない。配置された画面部品として例えばタブも同様であるとすると、タブ部分が操作されたとしても、操作されたタブに対応する画面（要素画面、表示領域）への切り替えは実行されない。したがって、例えばタブＡが選択され、タブＡに対応する要素画面が表示されている状態で、タブＢに対応する要素画面に対して他の画面部品を配置したい場合には、次のような操作を行う必要がある。
１つ目は、エディタ画面上でドラッグ＆ドロップするような操作ではなく、タブＢの要素画面に関連付けて他の要素部品を配置するような定義情報を記述または設定する操作を行うことである。しかしこれでは、ユーザーが定義情報の構造や記述方法、設定方法を理解している必要があり、容易ではない。
２つ目は、タブＢを直接操作するのとは異なる操作を行ってタブＢに対応する要素画面を表示させたうえで、他の画面部品の配置を行う必要がある。しかしこれでは操作手数の増大を招き煩わしい。また、直感的な操作方法でも無いため、ユーザーがタブをどのように切り替えればよいかわからない可能性がある。
画面部品が配置される要素画面を切り替えることが可能な画面部品（例えばドロワー）であれば、タブに限らず他の種別の画面部品についても同様である。これに対し、特許文献１、２には、画面部品を配置可能な要素画面を切り替えることが可能なタブのような画面部品による、切替対象となる要素画面に対して、どのような操作で他の画面部品を配置するかについては考慮されていない。
【０００６】
そこで本発明は、上記課題に鑑み、配置された画面部品による切替対象となる要素画面に対して、より容易に他の画面部品を配置可能な仕組みを提供する目的とする。
【課題を解決するための手段】
【０００７】
本発明の情報処理システムは、
アプリケーションの開発画面に配置された第１の種別のコンポーネントのうち指示領域に対する操作があったことに応じて、前記第１の種別のコンポ―ントの領域内の少なくとも一部の領域の表示を、操作された指示領域に対応する要素画面に切り替えるように制御する表示制御手段と、
前記開発画面に配置された前記第１の種別のコンポーネントの領域に対して他のコンポーネントを配置する操作があった場合に、前記第１の種別のコンポーネントの複数の要素画面のうち、現在表示されている要素画面に対して前記他のコンポーネントを配置するように制御する制御手段と
を有することを特徴とする。

【発明の効果】
【０００８】
本発明によれば、配置された画面部品による切替対象となる要素画面に対して、より容易に他の画面部品を配置することができる。
【図面の簡単な説明】
【０００９】
【図１】情報処理システムのシステム構成図である。
【図２】開発者用端末１００、アプリユーザー用端末２００、アプリユーザー用端末２０１のハードウェアブロック図である。
【図３】ログイン処理のフローチャートである。
【図４】ＵＩエディタ処理のフローチャートである。
【図５】ログイン処理とＵＩエディタ処理における表示例である。
【図６】ＵＩエディタ処理におけるタブ部品とＡｐｐＢａｒの表示例である。
【図７】コンテキストメニュー処理のフローチャートである。
【図８】アクションボード処理のフローチャートである。
【図９】アクションボード処理を説明するための表示例である。
【図１０】アクションボード処理における表示例である。
【図１１】開発環境３００が生成するソースコードの例である。
【図１２】画面切替処理のフローチャートである。
【図１３】アプリのユーザー情報の表示にかかる、開発者用端末１００、開発環境３００、実行環境４００においてそれぞれ実行される処理のフローチャートである。
【図１４】（ａ）開発者情報３０１の具体例である。（ｂ）マルチテナント実行環境のユーザー情報４１１の具体例である。（ｃ）シンブルテナント実行環境のユーザー情報の具体例である。
【図１５】ユーザー情報の表示例である。
【図１６】ワークフロー処理のフローチャートである。
【図１７】ワークフロー処理にかかる表示例である。
【図１８】ワークフロー処理によって生成されるデータベース、ＵＩ部品、アクションの表示例である。
【図１９】キャンバスのコンテキストメニュー処理のフローチャートである。
【図２０】ＣＲＵＤ生成処理にかかる表示例である。
【図２１】ＣＲＵＤ生成処理によって生成されるデータベース、アクション、関数の表示例である。
【図２２】開発者アカウント登録処理のフローチャートである。
【図２３】実行環境におけるＤＢセットの詳細である。
【図２４】デプロイ処理のフローチャートである。
【図２５】モバイル用アプリをデプロイする場合の表示例である。
【図２６】テンプレート画面に関するＵＩエディタ画面における表示例である。
【図２７】テンプレート画面のキャンバスにおける表示例である。
【図２８】プロパティボックスの表示例である。
【図２９】アクションが入力されたアクションボートの表示例である。
【図３０】データグリッドのコンテキストメニュー処理のフローチャートである。
【図３１】データグリッドのプロパティボックスにかかる表示例である。
【図３２】データグリッドのアクションボードにかかる表示例である。
【図３３】アクションの実行にかかるフローチャートである。
【図３４】アクションの情報を含むＵＩ定義情報の遷移図である。
【図３５】変形例におけるアクションの実行にかかるフローチャートである。
【図３６】変形例におけるアクションの情報を含むＵＩ定義情報の遷移図である。
【発明を実施するための形態】
【００１０】

以下、添付図面を参照して実施形態を詳しく説明する。尚、以下の実施形態は特許請求の範囲に係る発明を限定するものではなく、また実施形態で説明されている特徴の組み合わせの全てが発明に必須のものとは限らない。実施形態で説明されている複数の特徴のうち二つ以上の特徴が任意に組み合わされてもよい。また、同一若しくは同様の構成には同一の参照番号を付し、重複した説明は省略する。

【００１１】

以下に示す実施形態中で示した各種特徴事項は、互いに組み合わせ可能である。なお、以下において、「アプリケーション」、「アプリ」は、いずれも、アプリケーションソフトウェアを意味するものとする。

【００１２】

＜システム構成＞

図１に、本発明の実施形態としての情報処理システムのシステム構成図を示す。図１には、ソフトウェア開発を行うためのシステムと、開発されたソフトウェアを使用するためのシステムとを示す。

【００１３】

開発者用端末１００は、パーソナルコンピュータ（以下、ＰＣ）やスマートフォンなどで構成可能な開発者ユーザーが操作する情報処理装置（情報処理端末）である。すなわち、開発者ユーザーのユーザー端末である。開発者用端末１００は、開発するアプリケーションの設計操作を開発者から受け付け、設計された内容であるアプリケーションの各種定義情報を開発環境３００に送信する。

【００１４】

開発環境３００はネットワーク上（クラウド上、インターネット上）に構築された少なくとも１つのハードウェア資源を用いた環境である。開発環境３００はマルチテナントの環境であり、複数の開発者ユーザーがログイン可能な環境である。開発環境３００は、開発者情報３０１、実行エンジン３０２、ストレージ３２０を含む。開発環境３００は、複数のＷｅｂサービス（クラウドサービス）を組み合わせて構築される。

【００１５】

開発者情報３０１は、開発者のアカウントＩＤとなるメールアドレス（ユーザーＩＤ）、パスワードなどの、開発環境３００にログイン可能な開発者のアカウントを記録した情報である。開発者情報３０１は開発環境３００に含まれる少なくとも１つの記録媒体に記録されている。開発者情報３０１の詳細については図１４（ａ）を用いて後述する。

【００１６】

実行エンジン３０２は、開発環境３００で実行すべき処理を実行するための少なくとも１つのハードウェア資源であり、プロセッサ３０３とメモリ３０４を含む。プロセッサ３０３は少なくとも１つのプロセッサからなり、クラウド上の１つのプロセッサでもよいし、複数のプロセッサを組み合わせたプロセッサ群としても良い。メモリ３０４は、プロセッサ３０３が実行すべきプログラムを記録した少なくとも１つの記録媒体である。後述する各種フローチャートのうち、開発環境３００が実行するものとして説明するものは、実行エンジン３０２が実行する。すなわち、メモリ３０４に記録されたプログラムを、プロセッサ３０３が開発環境３００のうちワークメモリとなる領域に展開して実行することにより実現する。

【００１７】

配信エンジン３０５は、開発環境３００にアクセスした開発者用端末１００に対して、開発者用端末１００において実行すべきクライアント用プログラム３２２（ＨＴＭＬのソースコードやＪａｖａＳｃｒｉｐｔのソースコードなど）を送信する。クライアント用プログラム３２２は、開発環境３００に含まれるストレージ３２０に予め記録されている。

【００１８】

ストレージ３２０は、少なくとも１つの記録媒体の記憶領域であり、少なくとも、各開発者に共通であるクライアント用プログラム３２２を記憶している。また、開発者のアカウント毎の記憶領域である開発者用領域を有している。例えば、ログイン可能な開発者と

して開発者Ａ、開発者Ｂ、開発者Ｃがいる場合には、開発者Ａのための領域である開発者Ａ領域３２３、開発者Ｂのための領域である開発者Ｂ領域３２４、開発者Ｃのための領域である開発者Ｃ領域３２５が含まれる。各開発者用領域には、各開発者が開発したアプリの定義情報を記憶している。例えば、既発者Ａ領域３２３にはアプリ定義３２３ａ（図３４で後述するアプリのＵＩ定義情報、アプリの実行環境用プログラム含む）が記録されている。

【００１９】

開発者は、開発者用端末１００のブラウザソフトから、開発環境３００にアクセスするためのＵＲＬにアクセスすることで開発環境３００にアクセスし、開発環境にログインする。開発環境にログインすると、開発環境から、クライアント用プログラム３２２と、過去に開発作業を行って保存していた内容であるアプリ定義（ＵＩ定義情報）を受信する。そして、新規にアプリケーションを設計する操作、あるいは、既存の（作りかけの）アプリケーションの更新設計をする操作を行い、その結果であるアプリの定義情報（アプリ定義、ＵＩ定義情報）を開発環境３００に送信する。開発環境３００は受信したアプリ定義を、その開発者用領域に保存する。このように、本実施形態では、クラウド上の開発環境３００にアクセスすることが可能な端末であればどのような端末であっても開発者用端末１００として利用してアプリケーションの設計を行うことができる。従って、インターネットに接続することが可能な端末さえあれば、開発者は場所を選ばずにアプリケーションの開発を行うことができる。

【００２０】

実行環境４００は、ネットワーク上（クラウド上、インターネット上）に構築された少なくとも１つのハードウェア資源を用いた環境である。実行環境４００はマルチテナント実行環境４１０と、複数のシングルテナント実行環境（例えば、シングルテナント実行環境４５０、４６０、４７０）を含む。実行環境４００は、複数のＷｅｂサービス（クラウドサービス）を組み合わせて構築される。実行環境４００には、開発者用端末１００と開発環境３００を用いて開発されたアプリケーションの定義情報（アプリ定義）がデプロイされるための環境である。そして、アプリケーションを使用するユーザーが用いるアプリユーザー用端末２００、２０１は、アプリケーションの実行のためのＵＲＬにアクセスすることで、実行環境４００にアクセスする。そして、アプリユーザー用端末２００、２０１で行われた操作に応じた各種アクションを実行環境４００が実行することで、開発されたアプリケーションが実行され、アプリユーザーにアプリケーションの機能が提供される。ネットワーク上にある実行環境４００に構築されるアプリケーションはいわゆるＷＥＢアプリケーションである。

【００２１】

マルチテナント実行環境４１０は、複数の開発者に共用されるマルチテナント環境の実行環境であり、複数の開発者によって開発されたアプリケーションがデプロイされる。すなわち、複数の開発者によって共有される環境であり、複数の開発者の複数のアプリを構築可能な環境である。マルチテナント実行環境４１０には、ユーザー情報４１１、実行エンジン４１２、配信エンジン４１５、ストレージ４２０、ＤＢセット４３３を含む。

【００２２】

ユーザー情報４１１は、デプロイされたアプリケーション（構築されたアプリケーション）のユーザーアカウントＩＤとなるメールアドレス（ユーザーネーム）、パスワードなどの、アプリケーションにログイン可能なアプリユーザーのアカウントを記録した情報である。ユーザー情報４１１はマルチテナント実行環境４１０に含まれる少なくとも１つの記録媒体に記録されている。

【００２３】

実行エンジン４１２は、マルチテナント実行環境４１０で実行すべき処理を実行するための少なくとも１つのハードウェア資源であり、プロセッサ４１３とメモリ４１４を含む。プロセッサ４１３は少なくとも１つのプロセッサからなり、クラウド上の１つのプロセッサでもよいし、複数のプロセッサを組み合わせたプロセッサ群としても良い。メモリ４

１４は、プロセッサ４１３が実行すべきプログラムを記録した少なくとも１つの記録媒体である。後述する各種フローチャートのうち、マルチテナント実行環境４１０が実行するものとして説明するものは、実行エンジン４１２が実行する。すなわち、メモリ４１４に記録されたプログラムを、プロセッサ４１３がマルチテナント実行環境４１０のうちワークメモリとなる領域に展開して実行することにより実現する。ここで実行されるプログラムは、アプリケーションのアクションを実行するプログラムを含む。

【００２４】

配信エンジン４１５は、マルチテナント実行環境４１０にアクセスしたアプリユーザー用端末２００、２０１に対してアプリユーザー用端末２００、２０１において実行すべきクライアント用プログラム４２２（ＨＴＭＬのソースコードやＪａｖａＳｃｒｉｐｔのソースコードなど）を送信する。クライアント用プログラム４２２は、実行環境４１０に含まれるストレージ４２０に予め記録されている。

【００２５】

ストレージ４２０は、少なくとも１つの記録媒体の記憶領域であり、少なくとも、複数のアプリケーションで共通であるクライアント用プログラム４２２を記憶している。また、ストレージ４２０の所定の領域（所定のフォルダ、所定のバケット、所定の階層下）には、当該実行環境（マルチテナント実行環境４１０）へアクセスするためのアクセス先情報４２１が記録されている。また、開発者のアカウント毎の記憶領域である開発者用領域を有している。例えば開発者Ａのための領域である開発者Ａ領域４２３、開発者Ｂのための領域である開発者Ｂ領域４２４、開発者Ｃのための領域である開発者Ｃ領域４２５が含まれる。各開発者用領域には、各開発者が開発して、開発環境３００からデプロイされたアプリの定義情報を記憶している。例えば、既発者Ａ領域４２３にはアプリ定義４２３ａが記録されている。

【００２６】

ＤＢセット４３０は、実行環境４１０にデプロイされたアプリケーションが用いるデータベースに関する情報群である。ＤＢセット４３０は、少なくとも１つの記録媒体の記憶領域に記憶される。ＤＢセット４３０の詳細に関しては図２３を用いて後述する。

【００２７】

シングルテナント実行環境１（４５０）、２（４６０）、３（４７０）は、それぞれ１人の開発者（１つの開発者アカウント）に専用の実行環境であり、所有者である開発者によって開発環境３００を用いて開発されたアプリケーションがデプロイされる。本実施形態では例として、シングルテナント実行環境１（４５０）の所有者は開発者Ａ、シングルテナント実行環境２（４６０）の所有者も開発者Ａ、シングルテナント実行環境３（４７０）の所有者は開発者Ｂ、としている。このように、１人の開発者が複数のシングルテナント実行環境を所有することもできる。シングルテナント実行環境１（４５０）、２（４６０）、３（４７０）はそれぞれ、ユーザー情報４５１、４６１、４７１、実行エンジン４５２、４６２、４７２、配信エンジン４５５、４６５、４７５、ストレージ４５６、４６６、４７６、ＤＢセット４５７、４６７、４７７を含む。これらは、１人の開発者に専用であることを除いては、前述したマルチテナント実行環境４１０の、ユーザー情報４１１、実行エンジン４１２、配信エンジン４１５、ストレージ４２０、ＤＢセット４３３と同様の機能であるため、詳細な説明は省略する。シングルテナント実行環境は図示した３つだけでなく、更に数多く構築することが可能である。

【００２８】

図１のシステムを用いて、例えば、運用者が、マルチテナント実行環境４１０を開発者に無償提供し、シングルテナント実行環境を有償提供するという運用とすることが考えられる。マルチテナント実行環境４１０も、各シングルテナント実行環境も、リソースとサービスの提供業者（クラウドサービス事業者）に対して本システムの運用者が維持費用を支払う必要がある。マルチテナント実行環境４１０の維持費用は、運用者が負担して複数の開発者に無償提供することにより、開発者が本システムの試用のために費用負担をする必要がないため、多くの開発者が利用しやすく、本システムの普及を促進することができ

る。運用者は、シングルテナント実行環境に対して開発者に課金することで費用回収する。

【００２９】

１つの実行環境で単位時間あたりに処理できる処理リクエストの数に上限があり、数多くのアプリケーションがマルチテナント実行環境に構築され、多くのアプリユーザーが同時にアクセスした場合、リクエストが処理しきれず、アプリケーションの動作が遅くなるなどの状況になる可能性がある。他にも、マルチテナント実行環境に多くの開発者が開発した多くのアプリケーションがデプロイされて実行されることに対して、いくらかの制限があり、その制限のためにアプリケーションが十分なパフォーマンスを発揮できない場合がある。開発者が、有償で専用のシングルテナント実行環境を所有することで、このようなマルチテナント実行環境の制限による問題は回避（あるいは低減）することができる。すなわち、アプリケーションをシングルテナント実行環境に構築することで、十分なパフォーマンスを発揮するアプリケーションを構築することができる。

【００３０】

このようなマルチテナントシングル環境とシングルテナント実行環境の双方の特徴を踏まえ、開発者は、本システムを次のように活用できる。例えば、本システムを初めて使う場合にはマルチテナント実行環境４１０に本システムを用いて開発したアプリケーションを構築して試用を行った後に、本システムが開発者のソフトウェア開発に資すると判断したうえで、シングルテナント実行環境を有償で導入するといった使い方ができる。また、開発者が特定のアプリケーションＸを開発する際に、一般ユーザーに公開する前に、人数の限られたテストユーザーに試用してもらうためのアプリケーションＸのα版をマルチテナント実行環境４１０に構築する。そこでアプリケーションＸのテスト行い、アプリケーションＸの修正などを行い、更に開発を進める。そして、一般ユーザーへ公開しても問題ないレベルまで開発が完了した後に、アプリケーションＸの本運用版をシングルテナント実行環境に構築して、一般ユーザーに公開する。このような使い方をすれば、開発者は、開発期間の開発費用を抑え、かつ、多数の一般ユーザーが利用しても問題の無いアプリケーションの運用を行うことができる。

【００３１】

図２に、開発者用端末１００、アプリユーザー用端末２００、アプリユーザー用端末２０１として適用可能な装置（電子機器）の一例としての情報処理装置のハードウェアブロック図を示す。図２において、内部バス１５０に対してＣＰＵ１０１、メモリ１０２、不揮発性メモリ１０３、画像処理部１０４、ディスプレイ１０５、操作部１０６、記録媒体Ｉ／Ｆ１０７、外部Ｉ／Ｆ１０９、通信Ｉ／Ｆ１１０が接続されている。内部バス１５０に接続される各部は、内部バス１５０を介して互いにデータのやりとりを行うことができるようにされている。

【００３２】

メモリ１０２は、例えばＲＡＭ（半導体素子を利用した揮発性のメモリなど）からなる。ＣＰＵ１０１は、例えば不揮発性メモリ１０３に格納されるプログラムに従い、メモリ１０２をワークメモリとして用いて、情報処理装置の各部を制御する。不揮発性メモリ１０３には、画像データや音声データ、その他のデータ、ＣＰＵ１０１が動作するための各種プログラムなどが格納される。不揮発性メモリ１０３は例えばハードディスク（ＨＤ）やＲＯＭなどで構成される。

【００３３】

画像処理部１０４は、ＣＰＵ１０１の制御に基づいて、不揮発性メモリ１０３や記録媒体１０８に格納された画像データや、外部Ｉ／Ｆ１０９を介して取得した映像信号、通信Ｉ／Ｆ１１０を介して取得した画像データ、撮像された画像などに対して各種画像処理を施す。画像処理部１０４が行う画像処理には、Ａ／Ｄ変換処理、Ｄ／Ａ変換処理、画像データの符号化処理、圧縮処理、デコード処理、拡大／縮小処理（リサイズ）、ノイズ低減処理、色変換処理などが含まれる。画像処理部１０４は特定の画像処理を施すための専用の回路ブロックで構成しても良い。また、画像処理の種別によっては画像処理部１０４を

用いずにＣＰＵ１０１がプログラムに従って画像処理を施すことも可能である。
【００３４】
ディスプレイ１０５は、ＣＰＵ１０１の制御に基づいて、画像やＧＵＩ（Ｇｒａｐｈｉｃａｌ　Ｕｓｅｒ　Ｉｎｔｅｒｆａｃｅ）を構成するＧＵＩ画面などを表示する。ＣＰＵ１０１は、プログラムに従い表示制御信号を生成し、ディスプレイ１０５に表示するための映像信号を生成してディスプレイ１０５に出力するように情報処理装置の各部を制御する。ディスプレイ１０５は出力された映像信号に基づいて映像を表示する。なお、情報処理装置自体が備える構成としてはディスプレイ１０５に表示させるための映像信号を出力するためのインターフェースまでとし、ディスプレイ１０５は外付けのモニタ（テレビなど）で構成してもよい。以下、開発者用端末１００、アプリユーザー用端末２００、アプリユーザー用端末２０１が実行する処理においての表示先は、特に断りが無い場合には、各動作主のディスプレイ１０５であるものとする。
【００３５】
操作部１０６は、キーボードなどの文字情報入力デバイスや、マウスやタッチパネルといったポインティングデバイス、ボタン、ダイヤル、ジョイスティック、タッチセンサ、タッチパッドなどを含む、ユーザー操作を受け付けるための入力デバイスである。なお、タッチパネルは、ディスプレイ１０５に重ね合わせて平面的に構成され、接触された位置に応じた座標情報が出力されるようにした入力デバイスである。
【００３６】
記録媒体Ｉ／Ｆ１０７は、メモリーカードやＣＤ、ＤＶＤといった記録媒体１０８が装着可能とされ、ＣＰＵ１０１の制御に基づき、装着された記録媒体１０８からのデータの読み出しや、当該記録媒体１０８に対するデータの書き込みを行う。外部Ｉ／Ｆ１０９は、外部機器と有線ケーブルや無線によって接続し、映像信号や音声信号の入出力を行うためのインターフェースである。通信Ｉ／Ｆ１１０は、外部機器やインターネット１１１などと通信して、ファイルやコマンドなどの各種データの送受信を行うためのインターフェースである。開発者用端末１００は通信Ｉ／Ｆ１１０を用いて、インターネット１１１上にある開発環境３００と通信可能（情報の送受信可能）である。アプリユーザー用端末２００、２０１は通信Ｉ／Ｆ１１０を用いて、インターネット１１１上にある実行環境４００と通信可能（情報の送受信可能）である。
【００３７】
＜ログイン処理＞
図３（ａ）、（ｂ）に、ログイン処理のフローチャートを示す。この処理は、開発者用端末１００から開発環境３００にログインしてＵＩエディタを表示するまでの処理である。開発者用端末１００において、インターネットブラウザソフトを立ち上げ、本実施形態の開発システム（アプリケーション開発プラットフォーム）のＵＲＬを指定してアクセスする指示があると、図３（ａ）の処理を開始する。図３（ａ）の処理は、開発者用端末１００のＣＰＵ１０１が、インターネットブラウザソフトを実行するための不揮発性メモリ１０３に記録されたプログラムと、開発環境３００から受信したクライアント用プログラム３２２とを、メモリ１０２に展開して実行することにより実現する。以降、単に開発者用端末１００が実行する処理として記載したものは、開発者用端末１００のＣＰＵ１０１が、インターネットブラウザソフトを実行するための不揮発性メモリ１０３に記録されたプログラムと、開発環境３００から受信したクライアント用プログラム３２２とを、メモリ１０２に展開して実行する処理であるものとする。
【００３８】
開発者用端末１００において、インターネットブラウザソフトを立ち上げ、本実施形態の開発システム（アプリケーション開発プラットフォーム）のＵＲＬを指定してアクセスすると、開発環境３００の配信エンジン３０５がアクセスを検知し、アクセス元の開発者用端末１００にクライアント用プログラム３２２を送信する。
【００３９】
Ｓ３０１では、開発者用端末１００は、開発環境３００から送信されたクライアント用

プログラム３２２を受信したか否かを判定する。受信していなければＳ３０１で受信を待ち、受信したらＳ３０２に進む。

【００４０】

Ｓ３０２では、開発者用端末１００は、開発環境３００から受信したクライアント用プログラム３２２をメモリ１０２に記録する。

【００４１】

Ｓ３０３では、開発者用端末１００は、クライアント用プログラム３２２に従い、ログイン画面をディスプレイ１０５に表示する。ログイン画面には、本実施形態の開発システムへのログイン画面である旨と、開発者ＩＤとパスワードの入力欄、新規登録ボタン（アイコン）、ログインボタン（アイコン）が表示される。

【００４２】

Ｓ３０４では、開発者用端末１００は、ログイン画面の開発者ＩＤとパスワードの入力欄への入力操作を受け付ける。ユーザーの入力操作は操作部１０６を用いて行われる。開発者ＩＤは開発者ユーザーを識別（特定）するためのユーザー識別情報である。本実施形態では開発者ＩＤ（ユーザーネーム）は、メールアドレスであるものとする。また、暗証情報として用いるパスワードは任意の文字列であるものとするが、生体認証情報（指紋認証は顔認証に用いる情報）や、パターン認証の情報（画面に入力された軌跡パターンの情報）など、他の暗証情報としてもよい。

【００４３】

Ｓ３０５では、開発者用端末１００は、ログイン画面の新規登録ボタンを指示する操作（例えばクリック）されたか否かを判定する。なお、以下、表示アイテム（ボタン、アイコンなどの表示物、表示項目）に対して、操作部１０６に含まれるマウスでクリックする、タッチパネルへタッチする、といった方法で指示する操作を、単に「押下」と記載する。新規登録ボタンが押下された場合にはＳ３０６に進み、そうでない場合にはＳ３０７に進む。

【００４４】

Ｓ３０６では、開発者用端末１００は、開発者アカウント登録処理を行う。開発者アカウント登録処理の詳細は図２２（ａ）を用いて後述する。

【００４５】

Ｓ３０７では、開発者用端末１００は、ログイン画面のログインボタンが押下されたか否かを判定する。ログインボタンが押下された場合にはＳ３０８に進み、そうでない場合にはＳ３０４に戻る。

【００４６】

Ｓ３０８では、開発者用端末１００は、ログイン情報として、ログイン画面の開発者ＩＤとパスワードの入力欄へ入力された情報（入力された開発者ＩＤとパスワード）を開発環境３００に送信する。送信後、開発環境３００において認証処理が行われるため、その結果を待つ。

【００４７】

Ｓ３０９では、開発者用端末１００は、開発環境３００からログインエラーの旨の情報を受信したか否かを判定する。ログインエラーの旨の情報を受信した場合はＳ３０４に戻って再度ログイン情報の入力を受け付け、そうでない場合にはＳ３１０へ進む。

【００４８】

Ｓ３１０では、開発者用端末１００は、開発環境３００から実行環境リストを受信したか否かを判定する。開発環境３００はログイン認証が成功すれば実行環境リストを開発者用端末１００に送信するため、実行環境リストを受信したということは、ログイン認証に成功した（ログインできた）ということである。実行環境リストを受信した場合にはＳ３１１に進み、そうでない場合にはＳ３０９に戻る。

【００４９】

Ｓ３１１では、開発者用端末１００は、ログイン後のアプリの開発画面として、Ｓ３１０で受信した実行環境リストをメモリ１０２に記録するとともに、受信した実行環境リス

トに基づいて実行環境の選択肢をディスプレイ１０５に表示する。実行環境リストはログインした開発者がアクセス可能な実行環境を示している。なお、Ｓ３１１以降に開発者用端末１００で表示される画面であって、構築後のアプリの画面とは異なる画面を総称して開発画面と称するものとする。

【００５０】

図５（ａ）に、Ｓ３１１での実行環境の選択肢の表示例を示す。図５（ａ）の表示例では、ログインした開発者がアクセス可能な実行環境として、マルチテナント実行環境４１０に対応する選択肢５５１と、シングルテナント実行環境に対応する選択肢５５２の２つの選択肢が表示されている。開発者ユ―ザーはこれらの選択肢のいずれかを押下することにより選択して、ＳＡＶＥボタン５５３を押下することで選択を確定することができる。ここで選択されるのは、今回のログインにおいてこの後の作業で更新したアプリケーションをデプロイする先である。この時点で実行環境にアクセスするわけではない。また、ここで選択した実行環境は後述する操作によって変更可能である。

【００５１】

Ｓ３１２では、開発者用端末１００は、実行環境の選択が行われたか否かを判定する。実行環境の選択肢のいずれかが押下され、ＳＡＶＥボタン５５３が押下された場合にはＳ３１３に進み、そうでない場合にはＳ３１２で実行環境の選択を待つ。

【００５２】

Ｓ３１３では、開発者用端末１００は、Ｓ３１２で選択された実行環境を特定する情報（実行環境ＩＤなど）を「選択中実行環境」としてメモリ１０２に記録するとともに、開発環境３００に送信する。開発環境３００では、選択中実行環境（選択環境）を特定する情報を受信したことに応じて、アプリ情報として、ログイン開発者が所有する全てのアプリケーション（過去に生成され、ストレージ３２０に記録されているアプリケーション）を特定する情報（アプリＩＤやアプリ名など）を送信する。

【００５３】

Ｓ３１４では、開発者用端末１００は、開発環境３００からアプリ情報を受信したか否かを判定する。アプリ情報を受信した場合にはＳ３１５に進み、そうでない場合にはＳ３１４でアプリ情報の受信を待つ。

【００５４】

Ｓ３１５では、開発者用端末１００は、受信したアプリ情報をメモリ１０２に記録するとともに、アプリ情報に基づき、ログインユーザーが所有する（開発中の）アプリケーションの一覧（アプリ一覧）をディスプレイ１０５に表示する。

【００５５】

Ｓ３１６では、開発者用端末１００は、アプリ一覧に表示されたアプリケーションの一覧のうち、いずれかのアプリケーションが選択されたか否かを判定する。いずれかのアプリが選択された場合にはＳ３１７に進み、そうでない場合はＳ３２０に進む。

【００５６】

Ｓ３１７では、開発者用端末１００は、アプリ一覧から選択されたアプリケーションを「選択中アプリ」として、選択中アプリを特定する情報（アプリＩＤやアプリ名など）をメモリ１０２に記録するとともに、開発環境３００に送信する。開発環境３００では、選択中アプリを特定する情報を受信すると、ストレージ３２０のうち、ログインしている開発者の領域から選択中アプリの定義情報（アプリ定義）を取得し、開発者用端末１００に送信する。

【００５７】

Ｓ３１８では、開発者用端末１００は、開発環境３００から選択中アプリの定義情報（ＵＩ定義情報）を受信したか否かを判定する。選択中アプリの定義情報を受信した場合には、受信した選択中アプリの定義情報をメモリ１０２に記録してＳ３１９に進み、そうでない場合にはＳ３１８で定義情報の受信を待つ。本実施形態では、この定義情報は、Ｊｓｏｎ形式でアプリケーションに関する各種定義が記述されたＪｓｏｎファイルであるものとする。以降、選択中アプリに関してディスプレイ１０５に表示を行う場合には、メモリ

１０２に記録した定義情報に基づいて表示を行う。後述するＵＩエディタ処理などで選択中アプリについて更新する操作（例えば、ＵＩ部品の配置を変更するなど）が行われると、このメモリ１０２の定義情報を更新後の内容を定義するように更新する。そして保存の指示があった場合に、メモリ１０２に記録された最新の定義情報を開発環境３００に送信し、ストレージ３２０のうちログイン開発者の領域に保存させる。このようにすることで、開発環境３００との通信頻度の増大を抑え、通信のためにレスポンスが低下することを抑止して快適な更新作業を行うことができる。

【００５８】

Ｓ３１９では、開発者用端末１００は、ディスプレイ１０５にＵＩエディタ画面を表示するとともに、受信した定義情報に基づく表示を行う。例えば、選択中アプリがデスクトップ用であるか用であるかに応じた形状（すなわち、アプリを利用するデバイスの種別に応じた形状）のキャンバス（ＵＩ画面の編集領域）を表示する。テスクトップ用であれば１６：９の矩形のキャンバスとし、モバイル用であればスマートフォンを模した縦長のアスペクト比のキャンバスとする。サブメニュー領域（後述）には選択中アプリが有する（選択中アプリに属する）ＵＩ画面の一覧を表示する（この処理は厳密には後述する図４のＳ４０１で行われる）。また、キャンバスには、デフォルトで選択されるＵＩ画面（イニシャルＵＩや最後に保存した際に編集していたＵＩ画面）に配置されるコンポーネント（ＵＩ画面に配置されるＵＩ部品）を表示する。なお、編集対象のＵＩ画面をデフォルトで選択することをせず、この時点ではキャンバスには何も表示しないようにしても良い。Ｓ３１９の処理でログイン処理を終了し、続いて、図４のＳ４０１へ進む。

【００５９】

一方、Ｓ３２０では、開発者用端末１００は、アプリ一覧を表示した画面に表示された、アプリケーションを新規に作成するためのアイコン（＋マーク、不図示）が押下され、新規アプリケーションの作成が指示されたか否かを判定する。新規アプリケーションの作成が指示されたと判定した場合にはＳ３２１に進み、そうでない場合にはＳ３１６に戻る。

【００６０】

Ｓ３２１では、開発者用端末１００は、新規作成するアプリケーションを、デスクトップ用（ＰＣ用）とするかモバイル用とするかの選択画面を表示し、いずれかを選択する操作を受け付ける。デスクトップ用アプリケーションとは、デスクトップＰＣ、ノートＰＣなどのアプリユーザー用端末２００からアクセスされ、操作されることを想定したアプリケーションである。モバイル用アプリケーションとは、スマートフォンなどのアプリユーザー用端末２０１からアクセスされ、操作されることを想定したアプリケーションである。

【００６１】

Ｓ３２２では、開発者用端末１００は、新規作成するアプリケーションに関する基本となるアプリ情報（少なくともアプリ名、アプリＩＤ）の入力を受け付けるための画面を表示し、アプリ情報を設定する入力操作を受け付ける。アプリ情報の入力を受け付けると、Ｓ３２１で受け付けたデスクトップ用（ＰＣ用）かモバイル用かの情報と、Ｓ３２２で入力を受け付けたアプリ情報を開発環境３００に送信する。こうして、新規作成されたアプリケーションの定義情報として、開発環境３００のストレージ３２０に、新規のアプリケーションの定義情報が作成され、デスクトップ用（ＰＣ用）かモバイル用かの情報と、アプリ名、アプリＩＤが記録される。過去に作成済みのアプリケーションの定義情報にも、このようにして、デスクトップ用（ＰＣ用）かモバイル用かの情報と、アプリ名、アプリＩＤが記録されている。

【００６２】

Ｓ３２３では、開発者用端末１００は、新規作成されたアプリケーションの編集画面としてＵＩエディタ画面を表示する。この場合、キャンバスはＳ３２１で選択されたデスクトップ用であるかモバイル用であるかに応じた形状で表示される。また、キャンバスはコンポーネントが１つも配置されていない空白情報で表示される。Ｓ３２３の処理でログイ

ン処理を終了し、続いて、図４のＳ４０１へ進む。
【００６３】
図３（ｂ）に、図３（ａ）の開発者用端末１００側でのログイン処理と協働する開発環境３００側のログイン処理を示す。図３（ｂ）の処理は、開発環境３００のプロセッサ３０３が、メモリ３０４に記録されたプログラムを、プロセッサ３０３が開発環境３００のうちワークメモリとなる領域に展開して実行することにより実現する。以降、単に開発環境３００が実行する処理として記載したものは、開発環境３００の実行エンジン３０２が、より詳しくはプロセッサ３０３が実行する処理であるものとする。
【００６４】
Ｓ３３１では、開発環境３００は、Ｓ３０８で開発者用端末１００から送信されたログイン情報を受信したか否かを判定する。ログイン情報を受信した場合にはＳ３３２に進み、そうでない場合にはログイン情報の受信を待つ。
【００６５】
Ｓ３３２では、開発環境３００は、受信したログイン情報と、開発者情報３０１を比較し、ログイン認証（ユーザー認証）を行う。より具体的には、受信したログイン情報に含まれる開発者ＩＤとパスワードの組と一致する情報が、開発者情報３０１（ユーザー情報）に含まれているかを判定する。含まれていれば認証は成功する。
【００６６】
Ｓ３３３では、開発環境３００は、Ｓ３３２の認証処理の結果、ログインＯＫである（認証に成功した、認証された、認証ＯＫである）か否かを判定する。ログインＯＫである場合にはＳ３３５に進み、ログインＯＫでない場合にはＳ３３４に進み、ログインエラーである旨の情報を開発者用端末１００に送信する。
【００６７】
Ｓ３３５では、開発環境３００は、開発者情報３０１に含まれる、ログインＯＫとなった開発者（ログイン開発者）の実行環境リストを開発者用端末１００に送信する。開発者情報３０１には、図１４（ａ）に示す通り、各開発者について、メールアドレス（ユーザーネーム、開発者ＩＤ）とパスワードに加えて、アクセス可能な実行環境ＩＤが記録されている。各実行環境ＩＤはクラウドサービス（Ｗｅｂサービス）におけるアカウントＩＤであり、本実施形態では１２桁のＩＤであるものとする。複数の実行環境にアクセス可能な開発者の場合は、１２桁の実行環境ＩＤがカンマで区切られて記録されている。Ｓ３３５では、ログイン開発者についての、このアクセス可能な実行環境ＩＤ（カンマで区切られた１つ以上の実行環境ＩＤ）を開発者用端末１００に送信する。すなわち、Ｓ３３５において、開発者情報３０１を参照することにより、ログイン開発者がアクセス可能な実行環境が特定される。このように、各開発者のアクセス可能な実行環境（各開発者が利用可能な実行環境）は、開発環境３００に記録された開発者情報３０１に記録されている。そして、このログイン可能な実行環境は、ログインＯＫとなった開発者でなければ取得できない。また、ログインＯＫとなった開発者自身のアクセス可能実行環境しか取得できない。このようにすることで、開発者が、自身のアクセス可能な実行環境へアクセスするための情報を、開発環境３００にログインするための情報と別途に管理する必要がない。また、他のユーザーが不正に実行環境へアクセスすることも抑止できる。
【００６８】
Ｓ３３６では、開発環境３００は、Ｓ３１３で開発者用端末１００から送信された選択中実行環境を特定する情報（環境特定情報）を受信したか否かを判定する。選択中実行環境を特定する情報を受信した場合にはＳ３３７に進み、そうでない場合には選択中実行環境を特定する情報の受信を待つ。
【００６９】
Ｓ３３７では、開発環境３００は、Ｓ３３６で受信した選択中実行環境を特定する情報に基づいて、選択中実行環境を、ストレージ３２０のログイン開発者用の領域に保存されている、設定管理用のファイルに記録する。
【００７０】

Ｓ３３８では、開発環境３００は、ストレージ３２０のうち、ログイン開発者の領域から、ログイン開発者が所有する（ログイン開発者が作成した）全てのアプリケーションを示すアプリ情報を取得し、開発者用端末１００に送信する。ここで送信するアプリ情報は、アプリの定義情報のうち、Ｓ３１５で前述したアプリ一覧を表示するために必要な情報までであり、各アプリに関する詳細な定義情報（コンポーネントの配置や後述するアクションを示す情報など）は含まれない。

【００７１】

Ｓ３３９では、開発環境３００は、Ｓ３２２で開発者用端末１００から送信された新規アプリに関する情報（デスクトップ用（ＰＣ用）かモバイル用かの情報と、アプリ名、アプリＩＤを含む情報）を受信したか否かを判定する。新規アプリに関する情報を受信した場合にはＳ３４０に進み、そうでない場合にはＳ３４１へ進む。

【００７２】

Ｓ３４０では、開発環境３００は、ストレージ３２０のうち、ログイン開発者の領域に、Ｓ３３９で受信した新規アプリに関する情報に基づき、新規のアプリケーションの定義情報を作成して記録する。ここで記録される定義情報には、デスクトップ用（ＰＣ用）かモバイル用かの情報と、アプリ名、アプリＩＤが含まれる。なお、開発環境３００では、マルチテナント実行環境４１０において他のユーザーのアプリと区別するために、アプリＩＤとして、Ｓ３２２で開発者から入力されたＩＤの直前に、アプリを所有する開発者の開発者ＩＤに一意に対応する開発者コード８桁を付して記録する。そして、ログイン処理を終了する。以降、内部処理を行う場合とアクションボードにプログラミング言語で表示する場合にはアプリＩＤを用いる場合にはログイン開発者の開発者コードを付したＩＤで処理を行う。また、ＵＩエディタなどにアプリＩＤとして表示をする際には開発者コードを除く、Ｓ３２２で開発者から入力されたＩＤ部分だけを表示する。

【００７３】

Ｓ３４１では、開発環境３００は、Ｓ３１７で開発者用端末１００から送信された選択中アプリを特定する情報を受信したか否かを判定する。選択中アプリを特定する情報を受信した場合にはＳ３４２に進み、そうでない場合はＳ３３９に戻る。

【００７４】

Ｓ３４２では、開発環境３００は、Ｓ３４１で受信した選択中アプリを特定する情報に基づき、ストレージ３２０のうち、ログイン開発者の領域から、選択中アプリの定義情報（アプリ定義）を取得し、開発者用端末１００に送信する。ここで送信する定義情報は、選択中アプリに関する詳細な定義情報（コンポーネントの配置や後述するアクションを示す情報など）を含む。そして、ログイン処理を終了する。

【００７５】

＜ＵＩエディタ処理＞

図４、図５（ｂ）を用いて、ＵＩエディタ処理について説明する。ＵＩエディタ処理は、ＵＩエディタ画面（アプリケーション開発画面）に対する開発者（ユーザー）からの操作に応じて、構築するアプリケーションの各種定義（ＵＩ部品の定義、アクション定義）を行う処理である。

【００７６】

図５（ｂ）に、ディスプレイ１０５におけるＵＩエディタ処理で表示されるレイアウト編集画面の表示例を示す。図５（ｂ）の画面には、ヘッダーメニュー領域５００、メインメニュー領域５１０、サブメニュー領域５２０、キャンバス５３０（ＵＩ画面の編集受付領域）が含まれる。

【００７７】

ヘッダーメニュー領域５００には、選択中実行環境ボックス５０１、選択中アプリボックス５０２、選択中ＵＩ画面ボックス５０３、保存ボタン５０４、プレビューボタン５０５、デプロイボタン５０６が表示される。

【００７８】

選択中実行環境ボックス５０１には、選択中実行環境を表す情報として、選択中実行環

境ＩＤが表示される。選択中実行環境ボックス５０１の右端にある矢印アイコンを押下することで、プルダウンメニューとしてＳ３１０で取得したログイン開発者がアクセス可能な実行環境の一覧が表示され、一覧から任意の実行環境を選択することで、選択中実行環境を変更することが可能である。選択中実行環境が変更されても、選択中アプリは変更されず、メインメニュー領域５１０、サブメニュー領域５２０、キャンバス５３０に表示される内容は変更されない。このように、同じアプリケーションに関してデプロイする先である選択中実行環境を変更することで、同じアプリケーションを任意の複数の実行環境にデプロイすることが可能である。

【００７９】

選択中アプリボックス５０２には、選択中アプリを表す情報として、選択中アプリのアプリ名が表示される。選択中アプリボックス５０２の右端にある矢印アイコンを押下することで、プルダウンメニューとしてＳ３１４で取得したログイン開発者の所有するアプリ一覧が表示され、一覧から任意のアプリを選択することで、選択中アプリを変更することが可能である。選択中アプリが変更されると、サブメニュー領域５２０、キャンバス５３０に表示すべき内容が変わる。

【００８０】

選択中ＵＩ画面ボックス５０３には、キャンバス５３０で編集中のＵＩ画面を表す情報として、編集中のＵＩ画面名が表示される。選択中ＵＩ画面ボックス５０４にある矢印アイコンを押下することで、プルダウンメニューとして、Ｓ３１８で取得した選択中アプリの定義情報に基づき、選択中アプリに属するＵＩ画面の一覧が表示され、一覧から任意のＵＩ画面を選択することで、キャンバス５３０に表示する編集対象のＵＩ画面を変更することが可能である。

【００８１】

メインメニュー領域５１０には、メインメニューのメニュー項目としての選択肢アイコンとして、アプリ一覧ボタン５１１、ＵＩ画面ボタン５１２、ワークフローボタン５１３、設定ボタン５１４、環境一覧ボタン５１５、データベースボタン５１６、ファイルマネージャーボタン５１７、ユーザー管理ボタン５１８、スナップショットボタン５１９が表示される。これらの選択肢の押下に応じた処理については図１２の画面切替処理で後述する。

【００８２】

サブメニュー領域５２０には、メインメニューで選択された項目に応じたサブメニューが表示される。図５（ｂ）の例では、ＵＩ画面ボタン５１２の下位階層メニューとしてＵＩコンポーネント一覧（ＵＩ部品一覧）が表示されている例である。

【００８３】

キャンバス５３０は、選択中アプリの選択中のＵＩ画面（選択中ＵＩ画面ボックス５０３に表示されているＵＩ画面名のＵＩ画面）のレイアウト編集領域である。図５（ｂ）のキャンバス５３０は、デスクトップ用アプリケーションのキャンバスの表示例であり、デスクトップ用の形状で表示されている。ユーザーは、サブメニュー領域５２０に表示されたＵＩ部品一覧の中から任意のＵＩ部品（ＵＩコンポーネント）を選び、キャンバス領域５３０にドラッグアンドドロップで配置することができる。キャンバス領域５３０に配置されたＵＩ部品を選択してサイズや位置の調整ができる。また、キャンバス領域５３０に配置されたＵＩ部品を選択して右クリックして表示される右クリックメニュー（コンテキストメニュー）に含まれるプロパティを選択することで、配色などのより詳細な設定を行える。さらに、同じくコンテキストメニューに含まれるアクションを選択することで、アクションボードが表示され、そのＵＩ部品が操作された場合に実行すべきアクションを設定することができる。キャンバス５３０の空白領域にカーソルがある状態で右クリックを行うことで、キャンバスのコンテキストメニューを表示させることができ、そこに含まれるアクションを選択することにより、構築されたアプリケーションにおいてそのキャンバスのＵＩ画面がロードされた場合に実行（そのＵＩ画面を表示する際に実行）すべきアクションを設定することができる。

【００８４】
図５（ｂ）は、アプリ名「ＵＩ１」のアプリの、ＵＩ画面名「ｕｉ１」のキャンバス５３０に、ＵＩ部品として、パイチャート５３１（ｐｉｅ　ｃｈａｒｔ、円グラフ）、ボタン５３２、テキストフィールド５３３、５３４、アウトプットフィールド５３５、（Ｏｕｔｐｕｔ　Ｆｉｅｌｄ）、タブ部品５３６を配置した例である。操作パス５３１ａは、選択されているＵＩ部品を示す選択枠かつ拡大縮小の指示を受け付ける操作パス（操作ハンドル）であり、パイチャート５３１が選択されていることを示している。
【００８５】
図４に、ＵＩエディタ処理のフローチャートを示す。この処理は、開発者用端末１００が実行する処理であり、図３のＳ３１９またはＳ３２３の後に続けて行われる処理である。
【００８６】
Ｓ４０１では、開発者用端末１００は、選択中アプリの定義情報に基づき、サブメニュー領域５２０に選択中アプリのＵＩ画面一覧を選択肢として表示する。このＵＩ画面一覧に表示される各画面は、選択中アプリを実行環境にデプロイして構築し、アプリユーザー用端末２００、２０１からアクセスしてアプリを実行した場合に表示する画面として設計している画面である。このＵＩ画面一覧では、新規ＵＩ画面を追加する操作や、ＵＩ画面の削除操作も受付可能である。
【００８７】
Ｓ４０２では、開発者用端末１００は、Ｓ４０１で表示されたＵＩ画面一覧のうちいずれかのＵＩ画面を選択する操作があったか否かを判定する。いずれかのＵＩ画面が選択された場合はＳ４０４に進み、そうでない場合にはＳ４０３に進む。
【００８８】
Ｓ４０３では、メインメニュー領域５１０に表示されたいずれかの選択肢を選択する操作があったか否かを判定する。メインメニュー領域５１０に表示された選択肢が選択された場合にはＵＩエディタ処理を終了し、１１２で後述する画面切替処理へ進む。そうでない場合にはＳ４０２に戻る。
【００８９】
Ｓ４０４では、開発者用端末１００は、メモリ１０２に記録した選択中アプリの定義情報に基づいて、キャンバス５３０にＳ４０２で選択されたＵＩ画面を表示する。過去にＵＩ部品が配置済みのＵＩ画面であれば、定義情報に従って過去に配置されたＵＩ部品がキャンバス５３０に表示される。すなわち、過去に途中まで作成したＵＩ画面であれば続きから開発できる。Ｓ４０２で選択されたＵＩ画面が新規に作成されたＵＩ画面であれば、キャンバス５３０はＵＩ部品が配置されていない空白の状態で表示される。Ｓ４０２で選択されたＵＩ画面が、テンプレ―トとして予め用意されているＵＩ画面（テンプレート画面）である場合には、ユーザーがそのＵＩ画面に過去にＵＩ部品を配置していなくても、予め定められたアクションが定義された雛形コンポーネントがキャンバス５３０の予め決まった位置に表示される。
【００９０】
Ｓ４０５では、開発者用端末１００は、サブメニュー領域５２０に、ＵＩ部品の一覧を表示する。すなわち、選択中アプリのＵＩ画面一覧から、ＵＩ部品一覧の表示に切り替える。キャンバス５３０に配置可能なＵＩ部品としては、種別としてＩＮＰＵＴ、Ｂｕｔｔｏｎ（ボタン）、Ｄｉｓｐｌａｙ（情報表示用部品）、Ｎａｖｉｇａｔｉｏｎ、Ｌａｙｏｕｔ、Ｃｈａｒｔが含まれ、それぞれに種別に複数のＵＩ部品が分類されている。ＵＩ部品一覧では、まず、図５（ｃ）に図示するように、ＵＩ部品の種別の一覧が表示され、表示された種別のいずれかを選択する操作があったことに応じて、選択された種別に分類されるＵＩ部品が展開表示される。前述した図５（ｂ）は、ＩＮＰＵＴに対応する種別の選択肢５２２が選択され、ＩＮＰＵＴに分類されたＵＩ部品の一覧が表示された例である。ＩＮＰＵＴに分類されたＵＩ部品には、例えばテキストフィールド５２３、テキストエリア５２４が含まれる。サブメニュー領域５２０はスクロール可能であり、表示しきれない

選択肢（展開している種別のＵＩ部品の選択肢や、他の種別の選択肢）はスクロールして表示させることができる。図５（ｂ）のように展開している種別の選択肢５２２が操作されると、展開していた種別のＵＩ部品一覧が折りたたまれ、ＵＩ部品の種別の一覧が表示される。

【００９１】

Ｓ４０６では、開発者用端末１００は、サブメニュー領域５２０に表示されたＵＩ部品が選択されたか否かを判定する。より詳しくは、サブメニュー領域５２０に表示されたＵＩ部品をドラッグする操作が行われたか否かを判定する。サブメニュー領域５２０に表示されたＵＩ部品が選択された場合にはＳ４０７に進み、そうでない場合にはＳ４１１へ進む。

【００９２】

Ｓ４０７では、開発者用端末１００は、キャンバス５３０上の位置を指定する操作があったか否かを判定する。より詳しくは、ドラッグしていたＵＩ部品をキャンバス５３０上にドロップする操作があったか否か判定する。キャンバス５３０上を指定する操作があった場合はＳ４０８へ進み、そうでない場合にはＳ４０７でキャンバス５３０上の位置の指定を待つ。なお、本実施形態ではドラッグアンドドロップの例を説明するが、サブメニュー領域５２０からＵＩ部品を選択してキャンバス５３０上の指定位置に配置するための操作であれば操作方法はこれに限るものではない。

【００９３】

Ｓ４０８では、開発者用端末１００は、Ｓ４０７で指定されたキャンバス上の位置が、既に配置されているタブ部品（ＵＩ部品の一種）の領域に含まれるか否かを判定する。タブ部品の領域に含まれていない場合にはＳ４０９に進み、タブ部品の領域に含まれている場合にはＳ４１０へ進む。タブ部品は、例えば図５（ｂ）に示すタブ部品５３６である。タブ部品は複数のタブを有し（図５（ｂ）の例ではＩＴＥＭ１、ＩＴＥＭ２、ＩＴＥＭ３と表示された３つのタブ）、いずれかのタブが選択されたことに応じて、タブ部品で表示する表示内容を、選択されたタブに対応する要素画面に切り替える。

【００９４】

図６（ａ）、（ｂ）を用いて、タブ部品について説明する。図６（ａ）は、図５（ｂ）とは異なるＵＩ画面を表示したキャンバス５３０にタブ部品６０１を配置した場合のディスプレイ１０５における表示例である。タブ部品６０１の操作バス６０１ａで示される範囲が、タブ部品６０１の占有領域である。操作バス６０１ａを操作することで、要素画面を含むタブ部品６０１の全体の表示位置、全体の表示サイズを変更することができる。タブ部品６０１は、タブ６１０、タブ６２０、タブ６３０の３つのタブを持っている。各タブには、タブ部品のプロパティ設定で開発者によって設定可能なラベルがタブ名として表示されている。図示の例では、それぞれ、Ｔａｂ０、Ｔａｂ１、Ｔａｂ２と表示されている。タブの数、順序はタブ部品のプロパティ設定から変更可能である。タブのプロパティ設定は、図７で後述するコンテキストメニュー処理のＳ７０６において、タブのコンテキストメニューからタブのプロパティボックスを開く操作があり、それに応じて表示されるタブのプロパティボックス（設定画面）に対する設定操作によって行われる。一点破線（説明のために図示したものであって表示されるものではない）で示す要素画面領域６０２は、選択されたタブに応じて表示内容が切り替わる領域である。要素画面領域６０２に表示される各タブに対応する表示内容を、各タブに対応する要素画面と称する。各タブに対応する要素画面にはそれぞれ異なるＵＩ部品を配置可能である。図６（ａ）の例は、タブ６１０が選択され、タブ６１０に対応する要素画面が表示された例である。図示の例では、タブ６１０に対応する要素画面には、ＵＩ部品６１１とＵＩ部品６１２が配置されている。図６（ｂ）の例は、図６（ａ）の状態からタブ６２０がクリックされ、選択タブがタブ６１０からタブ６２０に変更された場合の表示例である。図６（ｂ）では、要素画面領域６０２には選択されているタブ６２０に対応する要素画面が表示されている。図示の例では、タブ６２０に対応する要素画面には、ＵＩ部品６２１が配置されている。図６（ａ）、図６（ｂ）の例では、定義情報には少なくとも、キャンバス５３０で編集中のＵＩ画

面のＩＤ（ＵＩ画面の識別情報）に対応付けて、タブ部品６０１のＵＩ部品ＩＤ（ＵＩ部品の識別情報）とＵＩ画面中におけるタブ部品６０１の位置が記録される。また、タブ部品６０１のタブ６１０のＩＤ（要素画面の識別情報）に対応付けて、ＵＩ部品６１１とＵＩ部品６１２のＩＤと、タブ６１０の要素画面中におけるＵＩ部品６１１とＵＩ部品６１２のそれぞれの位置が記録される。また、タブ部品６０１のタブ６２０のＩＤに対応付けて、ＵＩ部品６２１のＩＤと、タブ６２０の要素画面中におけるＵＩ部品６２１の位置が記録される。図４の説明に戻る。

【００９５】

Ｓ４０９では、開発者用端末１００は、サブメニュー領域５２０からＳ４０６で選択されたＵＩ部品を、キャンバス５３０上の指定位置にデフォルトのサイズで配置し、そのことを定義する情報をメモリ１０２に記録した定義情報に記録する。すなわち、定義情報において、配置先のＵＩ画面（編集中のＵＩ画面）のＩＤに、Ｓ４１０で配置したＵＩ部品の種別、ＵＩ部品ＩＤ、配置座標、配置サイズなどを関連付けて記録する。

【００９６】

Ｓ４１０では、開発者用端末１００は、サブメニュー領域５２０からＳ４０６で選択されたＵＩ部品を、キャンバス５３０上の指定位置のタブ部品のうち、選択中のタブに対応する要素画面（現在表示中の要素画面）にデフォルトのサイズで配置し、そのことを定義する情報をメモリ１０２に記録した定義情報に記録する。すなわち、定義情報において、配置先のＵＩ画面（編集中のＵＩ画面）のＩＤに、Ｓ４１０で配置したＵＩ部品の種別、ＵＩ部品ＩＤとしてタブ部品のＩＤ、および、そのタブ部品において選択中だったタブのＩＤ、選択中だったタブに対応する要素画面中における配置座標、配置サイズなどを関連付けて記録する。このようにすることで、本実施形態ではタブ部品の各タブに対応する各要素画面に、それぞれ異なるＵＩ部品を配置してレイアウトすることが可能である。またこのとき、ＵＩ部品の配置先の要素画面のタブを定義するために複雑な操作をする必要はなく、単に、配置したいＵＩ部品をドラッグアンドドロップで配置する際に、配置先の要素画面を選択して表示させておけばよい。

【００９７】

Ｓ４１１では、開発者用端末１００は、キャンバス５３０に配置済みのＵＩ部品がクリックなどで選択されたか否かを判定する。配置済みのＵＩ部品がクリックされた（選択された）場合にはＳ４１２に進み、そうでない場合にはＳ４１５に進む。

【００９８】

Ｓ４１２では、開発者用端末１００は、Ｓ４１１でクリックされたＵＩ部品に操作パスを表示する。操作パスの表示例が前述の操作パス５３１ａや操作パス６０１ａである。

【００９９】

Ｓ４１３では、開発者用端末１００は、Ｓ４１１においてクリックで指定された位置が、タブ部品のタブ部分の領域内であるか否かを判定する。タブ部分でない場合にはＳ４３１に進み、タブ部分である場合にはＳ４１４に進む。タブ部分は、対応する要素画面への切り替えを指示する指示領域である。

【０１００】

Ｓ４１４では、開発者用端末１００は、タブ部品の要素画面領域の表示を、Ｓ４１１においてクリックで指定された位置のタブに対応する要素画面に切り替える。例えば、Ｓ４１１でタブ部品５３６のうち、ＩＴＥＭ１、ＩＴＥＭ２、ＩＴＥＭ３と表示された３つのタブのいずれかの位置がクリックされたことに応じて、Ｓ４１２でタブ部品５３６に対して操作パスを表示するとともに、要素画面領域４３６ａの表示内容を、クリックされたタブに対応する要素画面のものに切り替える。また、例えば図６（ａ）のように表示されていた場合に、タブ６２０が指定されたことに応じて図６（ｂ）の表示に切り替える。すなわち、クリック前に表示されていた要素画面に配置されていたＵＩ部品６１１、６１２が非表示となり、代わりに、クリックされたタブ６２０に対応する要素画面に配置されているＵＩ部品６２１が表示される。このような制御をするのは、開発者がタブ部品の所望のタブの要素画面に他のＵＩ部品を配置したい場合に、配置先とする所望のタブの要素画面

を定義する操作を簡単にするためである。開発者は、タブ部品の所望のタブの要素画面に他のＵＩ部品を配置したい場合、配置したいＵＩ部品をドラッグアンドドロップで配置する前に、所望のタブを単にクリックして所望の要素画面を表示させておけばよい。

【０１０１】

なお、本実施形態では説明の簡略化のため、Ｓ４０９、Ｓ４１０、Ｓ４１３、Ｓ４１４の処理についてタブ部品を例にして説明したが、これに限るものではない。タブ部品以外でも、アプリケーションの構築後に、ＵＩ部品内の一部の表示内容が当該ＵＩ部品への操作に応じて切り替わる所定種別のＵＩ部品であれば、タブ部品と同様にＳ４０９、Ｓ４１０、Ｓ４１３、Ｓ４１４の処理を適用可能である。

【０１０２】

一方、タブ部品などの所定種別のＵＩ部品ではない他のＵＩ部品の場合には、キャンバス５３０上でＵＩ部品が選択されたことに応じて当該ＵＩ部品内の表示内容を切り替えることはしない。ＵＩ部品の選択に応じて行うのは操作バスの表示に係る処理だけであり、他のアクションは行わない。例えば、図５（ｂ）のキャンバス５３０でボタン５３２が選択された場合には操作パスが表示されるのみで、ボタン５３２のアクション（構築されたアプリでボタン５３２が押下された場合に実行される処理）は実行しない。また、キャンバス５３０でテキストフィールド５３３が選択された場合には操作パスが表示されるのみで、テキストフィールド５３３内に文字入力カーソルを表示するなどの処理は行われない。

【０１０３】

図６（ｃ）、図６（ｄ）、図６（ｅ）を用いて、タブ部品と同様にＳ４０９、Ｓ４１０、Ｓ４１３、Ｓ４１４の処理を適用可能な種別のＵＩ部品の例としてＡｐｐＢａｒの例を説明する。図６（ｃ）は、キャンバス５３０に、それぞれＵＩ部品であるＡｐｐＢａｒ６５０、ＴｅｘｔＦｉｅｌｄ６６１、Ｂｕｔｔｏｎ６６２を配置した場合の例である。ＡｐｐＢａｒ６５０は１つのＵＩ部品であり、ＡｐｐＢａｒ６５０の中に要素アイコン６５１と要素アイコン６５２が含まれる。要素アイコン６５１は、構築されたアプリケーションにおいては、ドロワーメニューを表示させるための指示を受け付ける表示アイテムである。要素アイコン６５２は、構築されたアプリケーションにおいては、ポップアップメニューを表示させるための指示を受け付ける表示アイテムである。キャンバス５３０において、配置済みのＡｐｐＢａｒ６５０のうち、要素アイコン６５１と要素アイコン６５２の位置がクリックされた場合にも、タブ部品のタブ部分と同様に制御する。より詳しくは以下のように制御する。

【０１０４】

配置済みのＡｐｐＢａｒ６５０のうち、要素アイコン６５１の位置がクリックされた場合（Ｓ４１１でＹｅｓ）、ＡｐｐＢａｒ６５０に対する操作パスを表示する（Ｓ４１２）とともに、Ｓ４１３でＹｅｓと判定され、要素画面としてのドロワーメニューを表示する（Ｓ４１４）。これによって、図６（ｃ）の表示状態から図６（ｄ）の表示状態へ遷移する。図６（ｄ）において、ＡｐｐＢａｒ６５０に対する操作パス６５０ａが表示され、かつ、要素アイコン６５１に対応する要素画面であるドロワーメニュー６５１ａが表示されている。ドロワーメニュー６５１ａは、画面左端から右側に引き出すようにして表示される領域であり、１つ以上のメニュー項目が表示される。ドロワーメニュー６５１ａを表示させた状態で、サブメニュー領域５２０から他のＵＩ部品をドラッグしてドロワーメニュー６５１ａにドロップすることで、タブ部品を例にしてＳ４０８、Ｓ４１０で説明した処理と同様、ドロワーメニュー６５１ａに対して他のＵＩ部品を配置可能である。

【０１０５】

配置済みのＡｐｐＢａｒ６５０のうち、要素アイコン６５２の位置がクリックされた場合（Ｓ４１１でＹｅｓ）、ＡｐｐＢａｒ６５０に対する操作パスを表示する（Ｓ４１２）とともに、Ｓ４１３でＹｅｓと判定され、要素画面としてのポップアップメニューを表示する（Ｓ４１４）。これによって、図６（ｃ）の表示状態から図６（ｅ）の表示状態へ遷移する。図６（ｅ）において、ＡｐｐＢａｒ６５０に対する操作パス６５０ａが表示され

、かつ、要素アイコン６５２に対応する要素画面であるポップアップメニュー６５２ａが表示されている。ポップアップメニュー６５２ａは、要素アイコン６５２の付近に表示される領域であり、１つ以上のメニュー項目（選択肢）が表示される。ポップメニュー６５２ａを表示させた状態で、サブメニュー領域５２０から他のＵＩ部品をドラッグしてポップアップメニュー６５２ａにドロップすることで、タブ部品を例にしてＳ４０８、Ｓ４１０で説明した処理と同様、ポップメニュー６５２ａに対して他のＵＩ部品を配置可能である。図４の説明に戻る。

【０１０６】

Ｓ４１５では、開発者用端末１００は、キャンバス５３０上で配置済みのＵＩ部品をドラッグする操作があったか否かを判定する。配置済みのＵＩ部品をドラッグする操作があった場合にはＳ４１６に進み、そうでない場合にはＳ４１７へ進む。Ｓ４１６では、開発者用端末１００は、ドラッグ操作に応じてドラッグされたＵＩ部品（選択部品）の配置される位置を変更する。具体的には、ドロップされた位置に配置する。配置を変更すると、メモリ１０２に記録している定義情報も変更後の位置を表すように更新する。

【０１０７】

Ｓ４１７では、開発者用端末１００は、キャンバス５３０上で配置済みのＵＩ部品の操作パスに対する操作があったか否かを判定する。操作パスへの操作があった場合にはＳ４１８に進み、そうでない場合にはＳ４１９へ進む。Ｓ４１８では、開発者用端末１００は、操作パスへの操作に応じて操作パスが付されたＵＩ部品（選択部品）のサイズを変更する。サイズを変更すると、メモリ１０２に記録している定義情報も変更後のサイズを表すように更新する。

【０１０８】

Ｓ４１９では、開発者用端末１００は、キャンバス５３０のいずれかの領域でコンテキストメニューを表示させる指示操作（本実施形態ではマウスの右クリック）があったか否かを判定する。右クリックがあった場合にはＳ４２０に進み、そうでない場合にはＳ４２１に進む。Ｓ４２０では、右クリックがあった際のマウスカーソルの位置に応じてコンテキストメニュー（右クリックメニュー）を表示し、コンテキストメニューに対する操作に応じた処理を行うコンテキストメニュー処理を行う。例えば、図５（ｂ）のボタン５３２上にマウスカーソルがある状態で右クリックが行われた場合には、図５（ｄ）に示すようなボタン５３２（指定ＵＩ部品）に関するコンテキストメニュー５４０を表示する。コンテキストメニュー５４０には、選択肢となるメニュー項目としてプロパティ５４１、アクション５４２、消去５４３が表示される。プロパティ５４１が選択されると、ボタン５３２に関するプロパティボックス（詳細設定ダイアログ）が表示され、ボタン５３２に表示するボタン名（ラベル）、ボタン５３２の色、数値指定でのサイズ、等の詳細な設定を行える。アクション５４２が選択されると、ボタン５３２に関するアクションボードが表示され、アクションボードに対してプログラミング言語であるＪａｖａＳｃｒｉｐｔでのアクションの入力が行える。ここで入力されるアクションは、構築されたアプリケーションにおいてボタン５３２（指定ＵＩ部品）が押下された場合に実行すべき処理である。消去５４３が選択されると、キャンバス５３０からボタン５３２を消去（削除）する。コンテキストメニュー処理の詳細については図７を用いて後述する。

【０１０９】

Ｓ４２１では、開発者用端末１００は、保存ボタン５０４が押下されたか否かを判定する。保存ボタン５０４が押下された場合にはＳ４２２に進み、そうでない場合にはＳ４２３へ進む。Ｓ４２２では、開発者用端末１００は、メモリ１０２に記録している編集中のアプリケーションの定義情報を開発環境３００に送信する。開発環境３００は、定義情報を受信すると図３３（ａ）または図３５（ａ）で後述する保存処理を行う。

【０１１０】

Ｓ４２３では、開発者用端末１００は、プレビューボタン５０５が押下されたか否かを判定する。プレビューボタン５０５が押下されたと判定した場合にはＳ４２４へ進み、そうでない場合にはＳ４２５へ進む。Ｓ４２４では、開発者用端末１００は、プレビュー処

理を行う。プレビュー処理では、キャンバス５０３を非表示とし、メモリ１０２に記録している定義情報または開発環境３００に記録されている定義情報に基づいて、キャンバス５０３で編集しているＵＩ画面について、構築されたアプリケーションでそのＵＩ画面を見た場合と同じ見た目となるようなプレビューを表示する。プレビューでは、ＵＩ部品を操作するための操作パスは表示されない。また、ＵＩエディタ画面では実行されない（キャンバス５３０への操作では実行されない）一部の動作について、プレビューでは実行される。例えば、画面遷移用のＵＩ部品やリンク部分を操作した場合に、画面遷移やリンク先への遷移が実行される。また、項目に対する選択枠の移動をキーボードのタブキーの操作によって行うといった動作も実行される。なお、アクションボートに開発者が入力したアクションやデータベースの参照といった動作は行われない。その分、プレビュー処理は実際にデプロイして確認する場合よりも高速に表示することができる。

【０１１１】

Ｓ４２５では、開発者用端末１００は、デプロイボタン５０６が押下されたか否かを判定する。デプロイボタン５０６が押下された場合にはＳ４２６へ進み、そうでない場合にはＳ４２７へ進む。Ｓ４２６では、開発者用端末１００は、デプロイ処理を実行する。デプロイ処理については図２４（ａ）を用いて後述する。なおデプロイボタン５０６が押下されてデプロイ処理を行う時には開発者はデプロイ先となる実行環境を選択する操作はする必要がなく、予め選択してあって、選択中実行環境ボックス５０１に表示されている選択中実行環境へデプロイが行われる。開発者は、特定の１つの実行環境にデプロイすべき内容をまとめて更新するように作業する場合が多い。そのため、本システムでは、更新すべきアプリケーションの選択より前にデプロイ先となる実行環境を選択させ（図３のＳ３１１）、アプリケーションをデプロイする際に都度実行環境を選択させることはない。従って、誤って意図しない実行環境にデプロイしてしまうという操作ミスを抑止し、また、作業を効率かすることができる。また、開発環境３００にログインする際に認証処理をするための操作を行ったものの、デプロイする実行環境に対して別途、アカウント認証処理するための操作をする必要は無い。従って作業手数の増大を防ぎ効率的に作業することが可能である。すなわち、開発者用端末１００は、デプロイ先の実行環境に関する認証情報を開発者ユーザーから取得することはない。

【０１１２】

Ｓ４２７では、開発者用端末１００は、選択中実行環境を変更する操作があったか否かを判定する。具体的には、選択中実行環境ボックス５０１に対して、選択中実行環境を変更する操作があったか否かを判定する。選択中実行環境を変更する操作があった場合にはＳ４２８へ進み、そうでない場合にはＳ４２９へ進む。Ｓ４２８では、開発者用端末１００は、選択された実行環境を特定する情報（実行環境ＩＤなど）を「選択中実行環境」としてメモリ１０２に記録するとともに、開発環境３００に送信する。また、選択中実行環境ボックス５０１の表示内容を、変更後の選択中実行環境を表すように更新する。開発環境３００では、選択中実行環境を特定する情報を受信すると、それに基づき、選択中実行環境を、ストレージ３２０のログイン開発者用の領域に保存されている、設定管理用のファイルに記録する。

【０１１３】

Ｓ４２９では、開発者用端末１００は、編集対象のアプリケーションを変更する操作があったか否かを判定する。具体的には、選択中アプリボックス５０２に対して選択中アプリを変更する操作があったか否かを判定する。選択中アプリを変更する操作があった場合には図３のＳ３１７に進み、変更後の選択中アプリに基づいてＳ３１７以降の処理を実行する。すなわち、サブメニュー領域５２０、キャンバス５３０の表示内容が更新される。選択中アプリを変更する操作がなかった場合には処理はＳ４３０へ進む。

【０１１４】

Ｓ４３０では、開発者用端末１００は、編集対象のＵＩ画面を変更する操作があったか否かを判定する。具体的には、選択中ＵＩ画面ボックス５０３に対して、選択中ＵＩ画面を変更する操作があったか否かを判定する。選択中ＵＩ画面を変更する操作があった場合

にはＳ４０４に進み、変更後の選択中ＵＩ画面に基づいて処理を行う。すなわち、キャンバス５３０の表示内容は変更後の選択中ＵＩ画面の内容に更新される（切り替わる）。選択中ＵＩ画面を変更する操作がなかった場合には処理はＳ４３１へ進む。

【０１１５】

Ｓ４３１では、開発者用端末１００は、メインメニュー領域５１０に表示されたいずれかの選択肢を選択する操作があったか否かを判定する。メインメニュー領域５１０に表示された選択肢が選択された場合にはＵＩエディタ処理を終了し、図１２で後述する画面切替処理へ進む。そうでない場合にはＳ４０６に戻る。

【０１１６】

＜コンテキストメニュー処理＞

図７に、コンテキストメニュー処理のフローチャートを示す。この処理は、図５のＳ４２０の詳細フローチャートである。

【０１１７】

Ｓ７０１では、開発者用端末１００は、表示すべきコンテキストメニューが、ボタン（Ｂｕｔｔｏｎ）の種別のＵＩ部品に関するものであるか否かを判定する。より詳しくは、Ｓ４１９で右クリックを受け付けた際にマウスで指定していたＵＩ部品（指定ＵＩ部品）の種別がボタンであるか否かを判定する。ボタンである場合にはＳ７１０に進み、そうでない場合にはＳ７０２に進む。

【０１１８】

Ｓ７０２では、開発者用端末１００は、表示すべきコンテキストメニューが、キャンバスに関するものであるか否かを判定する。より詳しくは、Ｓ４１９で右クリックを受け付けた際にマウスで指定していた位置が、キャンバス５３０のうちＵＩ部品が配置されていない空白領域であったか否かを判定する。空白領域であった場合（表示すべきコンテキストメニューが、キャンバスに関するものである場合）はＳ７０３に進み、そうでない場合にはＳ７０４に進む。

【０１１９】

Ｓ７０３では、開発者用端末１００は、キャンバスのコンテキストメニュー処理を行う。キャンバスのコンテキトメニュー処理については図１９を用いて後述する。

【０１２０】

Ｓ７０４では、開発者用端末１００は、表示すべきコンテキストメニューが、データグリッド（表）の種別のＵＩ部品に関するものであるか否かを判定する。より詳しくは、Ｓ４１９で右クリックを受け付けた際にマウスで指定していたＵＩ部品（指定ＵＩ部品）の種別がデータグリッドであるか否かを判定する。データグリッドである場合にはＳ７０５に進み、そうでない場合にはＳ７０６に進む。

【０１２１】

Ｓ７０５では、開発者用端末１００は、データグリッドのコンテキストメニュー処理を行う。データグリッドのコンテキトメニュー処理については図３０を用いて後述する。

【０１２２】

Ｓ７０６では、開発者用端末１００は、その他の処理として、指定された位置に対応する指定対象に関するコンテキストメニューを表示し、操作に応じた処理を行う。詳細は省略する。

【０１２３】

Ｓ７１０では、開発者用端末１００は、編集対象のＵＩ画面（選択中ＵＩ画面）がテンプレート画面であるか否かを判定する。テンプレート画面である場合にはＳ７２１に進み、そうでない場合にはＳ７１１に進む。

【０１２４】

Ｓ７１１では、開発者用端末１００は、指定ＵＩ部品に関するコンテキストメニューとして、図５（ｄ）に示したボタン用のコンテキストメニューを指定位置（マウスカーソル位置）の近傍に重畳表示する。

【０１２５】

Ｓ７１２では、開発者用端末１００は、コンテキストメニューに含まれる選択肢のうち、プロパティ（図５（ｄ）のプロパティ５４１）が選択されたか否かを判定する。プロパティが選択された場合にはＳ７１３に進み、そうでない場合にはＳ７１４へ進む。

【０１２６】

Ｓ７１３では、開発者用端末１００は、指定ＵＩ部品に関するプロパティボックス（詳細設定ダイアログ）として、ボタン用のプロパティボックスをキャンバス５３０に重畳して表示する。そして、プロパティボックスに対する各種設定操作を受け付ける。ここで、例えば、指定ＵＩ部品であるボタンに表示するボタン名（ラベル）、ボタンの色、数値指定でのサイズ、等の詳細な設定を行える。設定を行って反映操作を行うと、キャンバス５３０において、設定を反映した表示形態で指定ＵＩ部品が表示される。

【０１２７】

Ｓ７１４では、開発者用端末１００は、コンテキストメニューに含まれる選択肢のうち、アクション（図５（ｄ）のプロパティ５４２）が選択されたか否かを判定する。アクションが選択された場合にはＳ７１５に進み、そうでない場合にはＳ７１６へ進む。

【０１２８】

Ｓ７１５では、開発者用端末１００は、指定ＵＩ部品であるボタンに関するアクションボード処理を行う。アクションボード処理については図８を用いて後述する。

【０１２９】

Ｓ７１６では、開発者用端末１００は、コンテキストメニューに含まれる選択肢のうち、消去（図５（ｄ）の消去５４３）が選択されたか否かを判定する。消去が選択された場合にはＳ７１７に進み、そうでない場合にはＳ７１８へ進む。

【０１３０】

Ｓ７１７では、開発者用端末１００は、指定ＵＩ部品を消去し、メモリ１０２に記録した定義情報のうち、指定ＵＩ部品に関する情報（位置やサイズなどの定義、プロパティやアクションなど）を消去する。これによって、キャンバス５３０において指定ＵＩ部品は消去（削除）されて非表示となる。

【０１３１】

Ｓ７１８では、開発者用端末１００は、コンテキストメニューに含まれる選択肢のうち、他の選択肢が選択されたか否かを判定する。他の選択肢が選択された場合にはＳ７１９に進み、選択された選択肢に応じた処理を行う。そうでない場合にはＳ７２０へ進む。

【０１３２】

Ｓ７２０では、開発者用端末１００は、コンテキストメニューを閉じる操作（例えば、コンテキストメニューが表示された領域外をクリックする操作）があったか否かを判定する。コンテキストメニューを閉じる操作があった場合には、コンテキストメニューを非表示とし、図７の処理を終了する。閉じる操作がない場合にはＳ７１２に戻る。

【０１３３】

一方、Ｓ７２１では、開発者用端末１００は、指定ＵＩ部品に関するコンテキストメニューとして、図２６（ｄ）に示す、テンプレート画面用で、かつ、ボタン用のコンテキストメニューを指定位置（マウスカーソル位置）の近傍に重畳表示する。テンプレート画面用のコンテキストメニュー（図２６（ｄ）の雛形部品２６３４）には、通常のＵＩ画面用のコンテキストメニュー（図５（ｄ）に示したコンテキストメニュー５４０）と異なり、選択肢としてアクション、消去は表示されず、プロパティだけが表示される。すなわち、ＵＩ画面のうち、テンプレートの画面に配置されたＵＩ部品に関しては、当該ＵＩ部品が操作された場合のアクションを開発者が設定できないようにしている。この理由については後述する。

【０１３４】

Ｓ７２２では、開発者用端末１００は、コンテキストメニューに含まれる選択肢のうち、プロパティが選択されたか否かを判定する。プロパティが選択された場合にはＳ７２３に進み、そうでない場合にはＳ７２４へ進む。Ｓ７２３の処理はＳ７１３と同様である。

【０１３５】

Ｓ７２４では、開発者用端末１００は、コンテキストメニューを閉じる操作（例えば、コンテキストメニューが表示された領域外をクリックする操作）があったか否かを判定する。コンテキストメニューを閉じる操作があった場合には、コンテキストメニューを非表示とし、図７の処理を終了する。閉じる操作がない場合にはＳ７２２に戻る。

【０１３６】

＜アクションボード処理＞

アクションボードに関する処理について説明する。キャンバスに配置しているＵＩ部品を指定してコンテキストメニューからアクションを選択すると、指定されたＵＩ部品に関するアクションを設定するためのアクションボートを表示する。例えば、図９（ａ）に示すようなＵＩ画面を編集していた場合を考える。図９（ａ）は、ディスプレイ１０５におけるＵＩエディタ画面の一部を示す図である。図９（ａ）では、サブメニュー領域５２０にＵＩ部品一覧が表示され、キャンバス９００に編集対象のＵＩ画面が表示されている。キャンバス９００には、ボタンであるＵＩ部品９０１と、その他のＵＩ部品が配置されている。ＵＩ部品９０１が指定され、コンテキストメニューよりアクションの選択肢が選択されたことに応じて、図９（ｂ）に示すようなアクションボードが表示される。

【０１３７】

図９（ｂ）は、ＵＩ部品９０１に関するアクションボードの表示例である。アクションボード９１０は、キャンバス９００が表示されていた領域に、キャンバスに代えて表示される。テンプレート画面に予め配置される雛形部品（雛形コンポーネント）でなければ、開発者が過去に指定ＵＩ部品に対してアクションを設定していなければ、アクションボート９１０は図９（ｂ）のように空白で表示される。なお、アクションボード９１０に「１」と表示されているのは行数を示すガイド表示であって、アクションの内容ではない。開発者はこのアクションボード９１０に、操作部１０６に含まれるキーボードを操作して、プログラム言語であるＪａｖａＳｃｒｉｐｔで任意の文字列を入力することにより、任意のアクションを設定することができる。アクションボードで設定された内容を実行するトリガーは予め定められており、指定ＵＩ部品が操作されたことである。従って開発者は、アクションを実行するトリガーが何かというのは設定する必要がない（ＪａｖａＳｃｒｉｐｔで記述する必要はない）。例えば、指定ＵＩ部品がボタンである場合は、構築後のアプリにおいてそのボタンが押下されたことがトリガーであり、トリガーの発生に応じて、そのボタンのアクションボードに設定したアクションが実行される。

【０１３８】

また、アクションボード９１０とともに、サブメニュー領域には、ＵＩエディタ画面で表示されていた図９（ａ）のようなＵＩ部品一覧、あるいはＵＩ画面一覧に代えて、関数（Ｆｕｎｃｔｉｏｎ）一覧９２０が表示される。関数一覧９２０には、アクションボード９１０の表示を指示する選択肢９２２と、関数のそれぞれを表示させる指示をする関数の選択肢とが表示される。なお、予め用意された関数ではなく、開発者（ユーザー）が後述する関数追加ボタン９２１を押下して作成（追加）した関数を創作関数（作成関数、追加関数、自作関数）と称するものとする。自動的に作成された関数や予め用意された関数が無く、開発者が過去に指定ＵＩ部品に対して創作関数を作成していなければ、関数一覧９２０には、図９（ｂ）に示すように、関数の選択肢は表示されず、選択肢９２２と関数追加ボタン９２１だけが表示される。

【０１３９】

関数追加ボタン９２１が押下されると、創作関数を追加するためのダイアログボックス（追加ダイアログ９３０と称する）を表示する。図９（ｃ）に追加ダイアログ９３０の表示例を示す。追加ダイアログ９３０には、関数の種別（Ｆｕｎｃｔｉｏｎ　Ｔｙｐｅ）の選択を受け付ける種別選択欄９３１と、関数の名前（Ｆｕｎｃｔｉｏｎ　Ｎａｍｅ）の入力を受け付ける関数名入力欄９３２と、ＳＡＶＥボタン９３３が表示される。追加ダイアログに対する入力が行われ、ＳＡＶＥボタン９３３が押下されると、種別選択欄９３１で選択された種別に応じた関数の設定画面が表示されるとともに、関数一覧９２０に関数名入力欄９３２に入力された関数名が追加で表示される。

【０１４０】
図９（ｄ）は、種別選択欄９３１でＲＥＳＴが選択された場合に表示されるＲＥＳＴ関数設定画面の表示例である。アクションボード９１０に代えて、ＲＥＳＴ（ＲＥｐｒｅｓｅｎｔａｔｉｏｎａｌ　Ｓｔａｔｅ　Ｔｒａｎｓｆｅｒ）を用いた関数（ＲＥＳＴ関数）を作成するために必要な各種設定を受け付けるためのＲＥＳＴ関数設定画面９４０が表示される。また、関数一覧９２０には、関数の選択肢として、関数名入力欄９３２に入力された関数名「ｒｅｓｔ０１」の関数９２３が表示される。なお、「ｒｅｓｔ０１」の前に表示されているアイコンは、関数９２３の種別がＲＥＳＴ関数であることを示している。関数の設定画面で全ての必須設定項目が設定されていない場合（設定すべき情報が不足している場合）には、関数が未完成であることを示す未完マーク９２９が表示され、対応する関数９２３が未完成の状態であることが識別可能に表示される。

【０１４１】
図９（ｄ）のＲＥＳＴ関数設定画面９４０には、設定欄９４１～９４４が表示される。設定欄９４１は、関数名の設定画面であり、初期状態では関数名入力欄９３２に入力された関数名が表示されるが、開発者からの入力操作に応じて変更可能である。設定欄９４２はａｒｇｕｍｅｎｔｓ（引数）を設定するための変数名を設定する設定欄である。本実施形態ではｐａｒａｍがデフォルトで設定されており、変更できないものとする。変更できないことを示すためにグレーアウトして表示される。設定欄９４３は、ＲＥＳＴ関数のタイプを設定する設定欄である。この領域の右端の三角アイコンを操作することでプルダウンメニューに選択肢としてＧＥＴ、ＰＯＳＴ、ＰＵＴ、ＤＥＬＥＴＥが表示され、いずれかの選択肢を選択して設定することができる。ここで設定される種別は、この関数が行うリクエストの種別である。設定欄９４４はこの関数が行うリクエストを送信する先となるＵＲＬの設定欄である。このように、ＲＥＳＴ関数設定画面９４０でＲＥＳＴ関数を作成する場合には、開発者はプログラム言語でソースコードの文字列を記述する必要はない。

【０１４２】
図１０（ａ）～（ｄ）に、更に創作関数の設定画面とアクションボードの表示例を示す。図１０において、図９と同じものについては図１０と同じ符号を付し、説明を省略する。

【０１４３】
図１０（ａ）は、種別選択欄９３１でＳＱＬが選択された場合に表示されるＳＱＬ関数設定画面の表示例である。アクションボード９１０に代えて、ＳＱＬ（Ｓｔｒｕｃｔｕｒｅｄ　Ｑｕｅｒｙ　Ｌａｎｇｕａｇｅ）を用いた関数（ＳＱＬ関数）を作成するために必要な各種設定を受け付けるためのＳＱＬ関数設定画面９５０が表示される。また、関数一覧９２０には、関数の選択肢として、関数名入力欄９３２に入力された関数名「ｓｑｌ０１」の関数９２４が未完マーク９２９とともに表示される。なお、「ｓｑｌ０１」の前に表示されているアイコンは、関数９２４の種別がＳＱＬ関数であることを示している。設定欄９５１は、関数名の設定画面であり、初期状態では関数名入力欄９３２に入力された関数名が表示されるが、開発者からの入力操作に応じて変更可能である。設定欄９５２はａｒｇｕｍｅｎｔｓ（引数）を設定するための変数名を設定する設定欄である。本実施形態ではｐａｒａｍがデフォルトで設定されており、変更できないものとする。変更できないことを示すためにグレーアウトして表示される。設定欄９５３は、コンピュータ言語の一種であるＳＱＬ（データベース言語であり、プログラミング言語ではない）で、データベースへ指示を出す文字列（ＳＱＬ文、ＳＱＬステートメント）を入力するための入力欄である。開発者は、設定欄９５３に対して、操作部１０６に含まれるキーボードなどから、任意のＳＱＬ　ＳＴＡＴＥＭＥＮＴを入力することができる。このように、ＳＱＬ関数設定画面９５０でＳＱＬ関数を作成する場合には、開発者はプログラム言語でソースコードの文字列を記述する必要はない。

【０１４４】
図１０（ｂ）は、開発者が創作関数である関数９２３～９２５を作成した後に、アクションボード９１０に、開発者がある程度ＪａｖａＳｃｒｉｐｔでコード（文字列）を入力

（記述）した後に、「ｓｑ」と入力した段階の表示例である。本実施形態では、開発者ユーザーが文字列を入力した際に、入力された文字列が作成済みの創作関数の関数名（識別情報）と前方一致している場合には、コードアシスト欄９１１に、前方一致した関数名を入力候補の選択肢として表示する。図示の例では、入力された「ｓｑ」が、作成済みの創作関数である関数９２４の関数名「ｓｑｌ０１」と前方一致するため、選択肢として「ｓｑｌ０１」が表示される。ユーザーが、コードアシスト欄９１１に表示された「ｓｑｌ０１」を選択する操作を行うと、「ｌ０１」を入力する操作を行わなくとも、図１０（ｃ）に示すように、アクションボード９１０に「ｓｑｌ０１」が入力される。このコードアシスト機能により、開発者（ユーザー）が創作関数の関数名の文字の全てをキーボードで打たなくとも創作関数の関数名（識別情報）を入力することができ、操作手数の削減に寄与する。また、作成済みの創作関数の関数名に前方一致していればコードアシスト欄９１１にフルネームが表示され、一致していなければ（すなわち、その文字列で始まる関数が存在しなければ）コードアシスト欄９１１は表示されない。従って開発者は、自身が作成済みの創作関数の関数名として誤っていないことを確認して入力することができ、関数名の入力ミスを防止することができる。なお、前方一致する創作関数が複数ある場合には、コードアシスト欄９１１には、複数の選択肢が表示される。また、コードアシスト欄９１１に表示される創作関数の関数名は、アクションボード９１０の対象となっている指定ＵＩ部品（図示の例ではＵＩ部品９０１）について作成された関数であって、他のＵＩ部品について作成された創作関数については表示されない。従って、誤って他のＵＩ部品について作成した創作関数の関数名であって、指定ＵＩ部品の創作関数としては存在しない関数名をアクションボード９１０に記述してしまうミスも防止することができる。

【０１４５】

また、図１０（ｃ）に示す通り、アクションボード９１０とともに、関数一覧９２０には、開発者が作成した創作関数の関数９２３～９２５のそれぞれについて、関数の種別（文字列の前のアイコン）と、関数の名称が表示される。従って開発者ユーザーは、アクションボード９１０にアクションのコード（ＪａｖａＳｃｒｉｐｔでの文字列）を入力（記述）しながら、どうような種別のどのような名称の創作関数を作成済みであるかを確認することができる。そのため、アクションボード９１０に記述できる有効な創作関数の確認や管理の手間（例えば、手元にメモしておく、別の画面を開いて確認する、といった手間）を削減することができる。また、関数名の入力ミスの防止にも寄与する。関数一覧９２０に表示される関数は、アクションボード９１０の対象となっている指定ＵＩ部品（図示の例ではＵＩ部品９０１）について作成された関数であって、他のＵＩ部品について作成された創作関数については表示されない。従って、誤って他のＵＩ部品について作成した創作関数の関数名であって、指定ＵＩ部品の創作関数としては存在しない関数名をアクションボード９１０に記述してしまうミスも防止することができる。また、創作関数は、指定ＵＩ部品のアクションの定義の際にのみ有効であり、他のＵＩ部品のアクションの定義には影響を及ぼさない。従って、指定ＵＩ部品について作成する創作関数の関数名が、他のＵＩ部品について作成された創作関数と同じ関数名となった場合、アクションボード９１０にその関数名が記述されたとしても指定ＵＩ部品について設定された関数設定のみが反映され、他のＵＩ部品について設定された関数設定は反映されない。従って、他のＵＩ部品について作成済みの創作関数の関数名と同じ関数名を用いたとしてもエラーとなることはない。そのため、ユーザーは、他のＵＩ部品について作成済みの創作関数との関数名の重複を気にすることなく創作関数の関数名を決め、アクションボード９１０に入力することができる。すなわち、アプリケーションの開発全体において多数の創作関数を作成することになるが、それらは指定ＵＩ部品に関連付けて自動的に管理・整理され、コードアシスト欄９１１や関数一覧９２０に表示されるため、開発者による多数の創作関数の管理が非常に容易になる。従って関数名の確認に係る手間・時間を削減することができる。また、関数名の入力ミスも抑制できる。そのため、関数名の入力ミスを犯した場合のデバック作業（バグ取りの作業）も抑制することができる。以て、ソフトウェア開発（アプリケーション開発）に掛かる負荷・工数を削減し、より効率的なソフトウェア開発（アプリケ

ーション開発）を行うことができる。
【０１４６】
図１０（ｄ）は、アクションボード９１０にアクションのコードをプログラミング言語で入力し終えた場合の表示例である。図示の例では、３行目の点線９６０で示した部分に、創作関数である「ｒｅｓｔ０１」（関数一覧９２０の関数９２３）を用いている。図１０（ｄ）のアクションボード９１０に図示したコードの文字列では、創作関数であるｒｅｓｔ０１の詳細は定義されていない。また、開発者がｒｅｓｔ０１の設定をＲＥＳＴ関数設定画面９４０で行った際にも、プログラミング言語でのｒｅｓｔ０１の詳細の定義は行っていない。この状態のアクションの定義を含む定義情報を開発環境３００にアップロードして保存した場合、開発環境３００が、後述する図３３（ａ）または図３５（ａ）の保存処理において、実行環境用プログラムとして、アップロードされた定義情報から、関数の設定内容とアクションボード９１０に記述された文字列に基づき、プログラミング言語での関数の定義を含むプログラミング言語でのソースコードを作成する。このように、開発環境３００は、プログラミング言語で記述された創作関数の識別情報（関数名）が含まれる文字列を取得すると、関数設定画面で設定された情報に基づいて、創作関数の機能を含む文字列（アクションボードに記載されたアクション）で示される機能（アクション）を実行可能とする制御を行う。
【０１４７】
図１１に、図１０（ｄ）のアクションボード９１０に記載された文字列と、ＲＥＳＴ関数設定画面９４０で設定されたｒｅｓｔ０１の設定内容に基づいて、開発環境３００が生成するプログラミング言語でのソースコードの例を示す。なお、図示の例において、左端に記載した１～７５の数字は、行数を示すために図示したものであって、ソースコードの一部ではない。図１１の例は、プログラミング言語でのｒｅｓｔ０１の詳細の定義を含む７５行のソースコードである。しかし開発者自身は７５行もの入力をする必要はない。ＲＥＳＴ関数設定画面９４０での設定を行い、図１０（ｄ）のアクションボード９１０に図示した分量（７行）の文字列を記述すれば、図１１に示すソースコードの内容を定義できる。すなわち、ローコードで効率的な開発を行うことができる。
【０１４８】
図８に、アクションボード処理のフローチャートを示す。この処理は、図７のＳ７１５で前述したアクションボード処理の詳細フローチャートであり、図９（ｂ）～（ｄ）、図１０（ａ）～（ｄ）の表示例を用いて説明した動作となるように制御する処理である。
【０１４９】
Ｓ８０１では、開発者用端末１００は、ディスプレイ１０５にアクションボードを表示する。指定ＵＩ部品に対してアクションが定義されていなければ、図９（ｂ）で説明したような空白のアクションボート９１０を含む画面を表示する。指定ＵＩ部品に対して既にアクションが定義されていれば、図１０（ｄ）で説明したような、アクションの文字列が入力されたアクションボード９１０を含む画面を表示する。
【０１５０】
Ｓ８０２では、開発者用端末１００は、アクションボード９１０に対してアクションを記述する操作（記述操作）があったか否かを判定する。アクションを記述する操作とは、例えば、アクションボード９１０を選択した状態で操作部１０６に含まれるキーボードに対する操作や、タッチパネルに表示されるソフトキーボードをタッチする操作によって行われるテキスト入力操作である。アクションを記述する操作があった場合にはＳ８０３に進み、そうでない場合にはＳ８１０に進む。
【０１５１】
Ｓ８０３では、開発者用端末１００は、アクションを記述する操作によるアクションの入力操作を受け付け、アクションボード９１０に、入力された文字を表示する。
【０１５２】
Ｓ８０４では、開発者用端末１００は、Ｓ８０３で入力された文字と、その前までに入力された文字とによる文字列が、創作関数の関数名に前方一致する（すなわち、一部一致

する）か否かを判定する。前方一致する場合にはＳ８０５に進み、そうでない場合にはＳ８０８に進む。

【０１５３】

Ｓ８０５では、開発者用端末１００は、Ｓ８０４で一致すると判定された創作関数の関数名をコードアシスト欄に選択肢として表示する。これによって、図１０（ｂ）で説明したようにコードアシスト欄９１１が表示される。なお、コードアシスト欄９１１には、創作関数だけでなく、プログラミング言語で一般的に利用可能な関数名などについても、入力された文字列に前方一致するものを表示しても良い。

【０１５４】

Ｓ８０６では、開発者用端末１００は、コードアシスト欄に表示された選択肢のいずれかが選択されたか否かを判定する。選択肢のいずれかが選択された場合にはＳ８０７に進み、選択された選択肢の関数名をコードアシスト欄９１１に表示する。例えば、図１０（ｂ）の状態からコードアシスト欄９１１の選択肢が選択されたことに応じて、アクションボード９１０の５行目に示すように、「ｓｑｌ０１」と入力して表示する。選択肢のいずれも選択されなかった場合には処理はＳ８０８に進む。

【０１５５】

Ｓ８０８では、開発者用端末１００は、アクションを記述する操作が継続して行われたか否かを判定する。アクションを記述する操作が継続して行われた場合にはＳ８０３に進み、そうでない場合にはＳ８０２に進む。

【０１５６】

Ｓ８１０では、開発者用端末１００は、関数を追加する指示をする操作が行われたか否かを判定する。具体的には、関数追加ボタン９２１が押下されたか否かを判定する。関数追加ボタン９２１が押下された場合にはＳ８１１に進み、そうでない場合にはＳ８２０に進む。

【０１５７】

Ｓ８１１では。開発者用端末１００は、創作関数を追加するための追加ダイアログ９３０を表示し、追加ダイアログ９３０に対する入力を受け付ける。追加ダイアログで入力を受け付ける内容は図９（ｃ）を用いて前述したとおりである。追加ダイアログへの入力が行われ、ＳＡＶＥボタン９３３が押下されるとＳ８１２に進む。

【０１５８】

Ｓ８１２では、開発者用端末１００は、サブメニュー領域の関数一覧９２０に、追加ダイアログの関数名入力欄９３２に入力された関数名を追加で、未完マーク９２９を付した状態で表示する。

【０１５９】

Ｓ８１３では、開発者用端末１００は、追加ダイアログの種別選択欄９３１で選択された種別がスクリプトであるか否かを判定する。スクリプトである場合にはＳ８１４に進み、そうでない場合にはＳ８１５に進む。Ｓ８１４では、開発者用端末１００は、スクリプト関数の設定画面（不図示）を表示し、開発者ユーザーからの設定操作を受け付ける。

【０１６０】

Ｓ８１５では、開発者用端末１００は、追加ダイアログの種別選択欄９３１で選択された種別がＳＱＬであるか否かを判定する。ＳＱＬである場合にはＳ８１６に進み、そうでない場合（すなわち、追加ダイアログの種別選択欄９３１で選択された種別がＲＥＳＴである場合）にはＳ８１７に進む。Ｓ８１６では、開発者用端末１００は、ＳＱＬ関数設定画面を表示し、開発者ユーザーからの設定操作を受け付ける。ＳＱＬ関数設定画面で受け付ける設定内容は図１０（ａ）を用いて前述したとおりである。

【０１６１】

Ｓ８１７では、開発者用端末１００は、ＲＥＳＴ関数設定画面を表示し、開発者ユーザーからの設定操作を受け付ける。ＲＥＳＴ関数設定画面で受け付ける設定内容は図９（ｄ）を用いて前述したとおりである。

【０１６２】

Ｓ８１８では。開発者用端末１００は、それぞれの種別の関数設定画面において、必須項目として予め定められた設定項目への入力が完了したか（設定が完了したか）を判定する。必須項目への入力が完了した場合にはＳ８１９に進み、そうでない場合にはＳ８０２に進む。

【０１６３】

Ｓ８１９では、開発者用端末１００は、設定中の創作関数に関して関数一覧９２０に表示された関数名に対応付けて表示していた未完マークを非表示とする。これによってユーザー（開発者）は、設定中の創作関数が、アクションボード９１０で使える有効な設定状態であることを認識することができる。

【０１６４】

Ｓ８２０では、開発者用端末１００は、関数一覧９２０に表示されたいずれかの創作関数が選択（押下）されたか否かを判定する。関数一覧９２０で創作関数が選択された場合には、Ｓ８１３に進み、Ｓ８１４、Ｓ８１６、Ｓ８１７のいずれかで、選択された創作関数の種別に応じた設定画面を、既に設定済みの内容を反映して表示する。ユーザー（開発者）は、設定画面に対して設定内容の変更や追加の設定を行うことができる（すなわち、創作関数の編集を行うことができる）。

【０１６５】

Ｓ８２１では、開発者用端末１００は、保存の指示操作（例えば保存ボタン９１５の押下）があったか否かを判定する。保存の指示操作があった場合にはＳ８２２に進み、そうでない場合にはＳ８２３へ進む。

【０１６６】

Ｓ８２２では、開発者用端末１００は、現在までにアクションボード処理で行われた内容をメモリ１０２に保持した定義情報に記録するとともに、開発環境３００に送信する。

【０１６７】

Ｓ８２３では、開発者用端末１００は、関数一覧９２０に表示された、アクションボード９１０の表示を指示する選択肢９２２が押下されたか否かを判定する。選択肢９２２が押下された場合はＳ８０１に進み、指定ＵＩ部品のアクションボードを表示する。これによって、関数の設定画面を表示していた場合には、アクションボードの表示に切り替わる。選択肢９２２が押下されていない場合にはＳ８２４へ進む。

【０１６８】

Ｓ８２４では、開発者用端末１００は、アクションボードを閉じる操作（アクションボード処理を終了する操作）があったか否かを判定する。アクションボードを閉じる操作が無い場合にはＳ８０２に進んで処理を繰り返す。アクションボードを閉じる操作があった場合にはアクションボードを非表示とし、選択中ＵＩ画面のキャンバスの表示に切り替え、ＵＩエディタ処理へ戻る。

【０１６９】

＜画面切替処理＞

図１２に、画面切替処理のフローチャートを示す。この処理は、図５（ｂ）の表示例で前述したメインメニュー領域５１０に表示された選択肢の選択に応じた処理である。

【０１７０】

Ｓ１２０１では、開発者用端末１００は、アプリ一覧ボタン５１１が押下されたか否かを判定する。アプリ一覧ボタン５１１が押下された場合は図３のＳ３１５に進み、そうでない場合はＳ１２０２に進む。

【０１７１】

Ｓ１２０２では、開発者用端末１００は、ＵＩ画面ボタン５１２が押下されたか否かを判定する。ＵＩ画面ボタン５１２が押下された場合はＳ１２０３に進み、そうでない場合はＳ１２０４へ進む。Ｓ１２０３では、図４のＵＩエディタ処理を行う。

【０１７２】

Ｓ１２０４では、開発者用端末１００は、ワークフローボタン５１３が押下されたか否かを判定する。ワークフローボタン５１３が押下されたと判定した場合はＳ１２０５に進

み、そうでない場合はＳ１２０６に進む。
【０１７３】
Ｓ１２０５では、開発者用端末１００は、ワークフロー作成処理を行う。ワークフロー作成処理は図１６を使って後述する。
【０１７４】
Ｓ１２０６では、開発者用端末１００は、設定ボタン５１４が押下されたか否かを判定する。設定ボタン５１４が押下された場合にはＳ１２０７に進み、そうでない場合はＳ１２０９へ進む。
【０１７５】
Ｓ１２０７では、開発者用端末１００は、アプリ設定画面を表示する。アプリ設定画面は、選択中アプリに関する設定操作を受け付ける画面である。Ｓ１２０８では、開発者用端末１００は、アプリ設定画面への設定操作を受け付け、設定に基づいてメモリ１０２に記録した定義情報を更新する。Ｓ１２０８で受け付け可能な設定には、例えば、表示言語の設定や、ＰＷＡ（Ｐｒｏｇｒｅｓｓｉｖｅ　Ｗｅｂ　Ａｐｐｓ）としての利用を可とするアプリを構築するか否かの設定がある。また、アプリに所属する複数のＵＩ画面のうち、どのＵＩ画面をイニシャルＵＩとするかの設定がある。イニシャルＵＩとは、実行環境にデプロイされた構築済みアプリにアクセスした際に最初に表示される画面、あるいは構築済みアプリにアクセスした後にアプリの認証画面において認証ＯＫになった後に最初に表示される画面である。
【０１７６】
Ｓ１２０９では、開発者用端末１００は、環境一覧ボタン５１５が押下されたか否かを判定する。環境一覧ボタン５１５が押下された場合には図３のＳ３１１に進み、そうでない場合にはＳ１２１０へ進む。
【０１７７】
Ｓ１２１０では、開発者用端末１００は、データベースボタン５１６が押下されたか否かを判定する。データベースボタン５１６が押下された場合にはＳ１２１１に進み、そうでない場合にはＳ１２１５へ進む。データベ―スボタン５１６の押下はデータベースへの接続要求である。
【０１７８】
Ｓ１２１１では、開発者用端末１００は、選択中実行環境がマルチテナント実行環境４１０であるか否かを判定する。マルチテナント実行環境４１０である場合にはＳ１２１３に進み、そうでない場合（シングルテナント実行環境である場合）にはＳ１２１２に進む。
【０１７９】
Ｓ１２１２では、開発者用端末１００は、開発環境３００が選択中実行環境から取得したデータベースの情報を取得する。より詳しくは、開発環境３００が、選択中実行環境がマルチテナント実行環境であるかシングルテナント実行環境であるかを判定し、シングルテナント実行環境である場合には、開発環境３００が選択中実行環境であるシングルテナント実行環境のＤＢセット（ＤＢセット４５７、４６７、４７７などのうちいずれか）にアクセスし、データベースに記録されている内容を取得する。そして、開発環境３００が選択中実行環境から取得したデータベースの情報を開発者用端末１００に送信する。開発者用端末１００から、開発環境３００を介さずにシングルテナント実行環境にアクセスすることは無い。
【０１８０】
Ｓ１２１３では、開発者用端末１００は、ログイン開発者の開発者情報（開発者情報３０１のうち、ログイン開発者について登録された情報）に記録されたＤＢインスタンス名を参照する。そして、開発環境３００が選択中実行環境であるマルチテナント実行環境４１０に含まれるＤＢセット４３０のうち、ログイン開発者の開発者情報を参照して得たＤＢインスタンス名が示すＤＢインスタンスのデータベースに記録されている内容を取得する。そして、開発環境３００が選択中実行環境から取得したデータベースの情報を開発者

用端末１００に送信する。開発者用端末１００から、開発環境３００を介さずにマルチテナント実行環境４１０にアクセスすることは無い。

【０１８１】

Ｓ１２１４では、開発者用端末１００は、データベースの管理画面を表示し、Ｓ１２１２またはＳ１２１３で取得したデータベースの情報を表示する。そして、選択中実行環境のデータベースに関する各種設定や、内容の更新の指示を行うＤＢ管理操作を受け付ける。

【０１８２】

Ｓ１２１５では、開発者用端末１００は、ファイルマネージャーボタン５１７が押下されたか否かを判定する。ファイルマネージャーボタン５１７が押下された場合にはＳ１２１６に進み、そうでない場合にはＳ１２１８に進む。

【０１８３】

Ｓ１２１６では、開発者用端末１００は、選択中実行環境に保存するファイルを管理するファイル管理画面を表示し、Ｓ１２１７で、選択中実行環境に保存するファイルを管理する操作を受け付ける。例えば、構築されるアプリケーションの画面に表示すべき画像ファイルを選択中実行環境の開発者領域にアップロードして保存することができる。開発者用端末１００のローカル保存領域（記録媒体１０８など）から選択中実行環境に保存すべきファイルが選択され保存の指示がされると、開発者用端末１００は、選択されたファイルを開発環境３００に送信する。開発環境３００は、選択されたファイルを受信すると、選択中実行環境の開発者領域に送信することで、選択中実行環境に保存させる。各種実行環境は、開発環境３００からのアクセスだけを受け付ける。そのため、ファイル管理のためには、開発者用端末１００から、開発環境３００を介して各種実行環境にアクセスする。開発者用端末１００から、開発環境３００を介さずにマルチテナント実行環境４１０にアクセスすることは無い。前述のログイン処理で説明した取り、開発者は、開発環境３００へは認証を行ってログインする必要がある。従って開発環境３００を介してでないと各種実行にアクセスできないようにすることで、開発環境３００にログインできない他のユーザーが不正に実行環境にアクセスすることは出来ず、その分、セキュリティが向上する。

【０１８４】

また、開発者ユーザーは開発環境３００にログインするための認証に掛かる操作だけを行えばよく、実行環境毎にログインするための認証にかかる操作をする必要がない。そのため、操作手数の増大を抑制することができる。

【０１８５】

Ｓ１２１８では、開発者用端末１００は、ユーザー管理ボタン５１８が押下されたか否かを判定する。ユーザー管理ボタン５１８が押下されたと判定した場合はＳ１２１９に進み、そうでない場合はＳ１２２０に進む。

【０１８６】

Ｓ１２１９では、開発者用端末１００は、ユーザー情報表示処理を行う。ユーザー情報表示処理は、選択中実行環境に構築されるアプリケーションにログイン可能なアプリユーザー（開発者とは別途管理される情報）の情報であるユーザー情報（図１に図示した各実行環境におけるユーザー情報４１１、４５１、４６１、４７１など）を表示し、開発者からの管理操作を受け付ける処理である。ユーザー情報表示処理については図１４を用いて後述する。

【０１８７】

Ｓ１２２０では、開発者用端末１００は、スナップショットボタン５１９が押下されたか否かを判定する。スナップショットボタン５１９が押下された場合にはＳ１２２１に進んでスナップ処理を行い、そうでない場合にはＳ１２２２に進む。

【０１８８】

Ｓ１２２２では、開発者用端末１００は、終了操作が行われたか否かを判定する。終了操作が行われていない場合にはＳ１２０１に進んで処理を繰り返し、終了操作があった場

合には画面切替処理を終了する。
【０１８９】
＜ユーザー情報表示処理＞
図１４（ａ）に、開発環境３００に保持している開発者情報３０１の具体例を示す。開発者情報３０１には、図示の各行の通り、開発者毎のアカウント情報として次の情報が開発者毎に関連付けて記録されている。開発者のアカウントＩＤ（開発者ＩＤ）となるメールアドレス（ユーザーネーム）、パスワード、氏、名、アクセス可能な実行環境ＩＤ、Ｖｅｒｉｆｙ、マルチテナント実行環境でのアクセス可能ＤＢインスタンス名。アクセス可能な実行環境ＩＤは、本実施形態ではクラウドサービスのアカウントＩＤであるものとする。１人の開発者が複数の実行環境にアクセス可能である場合、アクセス可能な実行環境ＩＤには、複数の実行環境ＩＤがカンマで区切って連続的に記載された文字列が記録される。Ｖｅｒｉｆｙは有効なアカウントであうかどうかを示す情報である。なお、１つのアカウントに関連付けて記録する情報は、図１４（ａ）に示す情報以外にあっても良い。
【０１９０】
図１４（ｂ）に、マルチテナント実行環境４１０に保持しているユーザー情報４１１の具体例を示す。ユーザー情報４１１には、図示の各行の通り、アプリのユーザー毎のアカウント情報として次の情報がアプリユーザー毎に関連付けて記録されている。アプリのユーザー毎のアカウント情報は、アプリユーザーのアカウントＩＤとなるメールアドレス（ユーザーネーム）、パスワード、氏、名、Ｅメールの承認済みであるか否かの情報、オーナーＩＤを含む。マルチテナント実行環境には、複数の開発者をオーナーとする複数のアプリケーションが混在してデプロイされている。そのため、アプリの各ユーザーがどの開発者（オーナー）の作ったアプリのユーザーであるかを識別するために、オーナーＩＤとして開発者ＩＤも記録されている。なお、１つのアカウントに関連付けて記録する情報は、図１４（ｂ）に示す情報以外にあっても良い。例えば、生成日時、更新日時などが記録されていてもよい。
【０１９１】
図１４（ｃ）に、シングルテナント実行環境に保持しているユーザー情報（ユーザー情報４５１、４６１、４７１など）の具体例を示す。シングルテナント実行環境のユーザー情報には、図１４（ｃ１）に示すユーザー別情報と、図１４（ｃ２）に示すグループ情報の２種類が記録される。
【０１９２】
図１４（ｃ１）に示すユーザー別情報には、図示の各行の通り、アプリのユーザー毎のアカウント情報として次の情報がアプリユーザー毎に関連付けて記録されている。アプリのユーザー毎のアカウント情報は、アプリユーザーのアカウントＩＤとなるメールアドレス（ユーザーネーム）、パスワード、氏、名、Ｅメールの承認済みであるか否かの情報を含む。一方、シングルテナント実行環境では、複数の開発者をオーナーとする複数のアプリケーションが混在することは無いため、オーナーＩＤは記録されない。なお、１つのアカウントに関連付けて記録する情報は、図１４（ｃ１）に示す情報以外にあっても良い。例えば、生成日時、更新日時などが記録されていてもよい。
【０１９３】
図１４（ｃ２）に示すグループ情報には、図示の各行の通り、複数のグループＩＤのそれぞれに対して、１人以上の所属ユーザーのユーザーネームが関連付けて記録されている。
【０１９４】
図１３（ａ）に、開発者用端末１００におけるユーザー情報表示処理のフローチャートを示す。この処理は、前述した図１２のＳ１２１８の詳細フローチャートである。
【０１９５】
Ｓ１３０では、開発者用端末１００は、開発環境３００のユーザー情報取得指示（選択中実行環境のユーザー情報の取得を指示する旨の情報）を送信する。
【０１９６】

Ｓ１３０２では、開発者用端末１００は、Ｓ１３０１で送信したユーザー情報取得指示に応じた情報であって、開発環境３００から送信されたユーザー情報を受信したか否かを判定する。ユーザー情報を受信していない場合にはＳ１３０２でユーザー情報の受信を待ち、ユーザー情報を受信した場合にはＳ１３０３に進む。取得されるユーザー情報は、選択中実行環境がマルチテナント実行環境であれば図１４（ｂ）に示したような情報であり、選択中実行環境がシングルテナント実行環境であれば図１４（ｃ１）、図１４（ｃ２）に示したような情報である。

【０１９７】

Ｓ１３０３では、開発者用端末１００は、Ｓ１３０２で受信したユーザー情報に基づいて、ディスプレイ１０５にユーザー情報を表示する。図１５に、Ｓ１３０３で表示される、開発者用端末１００におけるユーザー情報の表示例を示す。図１５において、前述した図５（ｂ）の表示例と同じ表示項目については同じ符号を付して説明を省略する。表示アイテム１５０１は、図１４（ｃ１）に示すユーザー別情報に基づいた表示を指示するための指示アイテムである。表示アイテム１５０２は、図１４（ｃ２）に示すグループ情報に基づいた表示を指示するための指示アイテムである。表示アイテム１５０１、１５０２はいずれかを切り替えて選択可能である。また、選択中実行環境がマルチテナント実行環境であった場合には表示アイテム１５０１、１５０２は表示されず、グループ情報への表示へは切り替えられないようになっている。グループ情報１５０５は、Ｓ１３０２で受信したユーザー情報（図示の例では、図１４（ｃ１）に示したユーザー別情報に基づいてユーザー情報を表示した例である。グループ情報１５０５において、Ｕｓｅｒｎａｍｅ、Ｅｍａｉｌ　Ｖｅｒｉｆｉｅｄの列はそれぞれ、図１４（ｃ１）に示したユーザー別情報におけるメールアドレス（ユーザーネーム）、Ｅメールの承認済みであるか否かの列の情報に対応する。Ｄｅｌｅｔｅの列には、各行に、その行のユーザーアカウントを削除する指示を受け付けるボタンアイコンが表示される。ユーザー追加ボタン１５０３は、グループ情報１５０５に１行追加して新たなアプリユーザーの情報の登録を指示する操作アイコンである。保存ボタン１５０４は図１５の画面で編集された内容のグループ情報１５０５を、選択中実行環境へ保存する指示を行うための操作アイコンである。

【０１９８】

Ｓ１３０４では、開発者用端末１００は、Ｓ１３０３で表示したユーザー情報の編集操作を受け付ける。例えば、ユーザー追加ボタン１５０３の押下によるユーザーアカウントの追加や、Ｄｅｌｅｔｅの列に表示されたボタンアイコンの押下によるユーザーカウントの削除、キーボードを操作しての編集可能項目に対する内容の編集などである。

【０１９９】

Ｓ１３０５では、開発者用端末１００は、保存ボタン１５０４が押下されたか否かを判定する。保存ボタン１５０４が押下された場合はＳ１３０６に進み、そうでない場合にはＳ１３０７へ進む。

【０２００】

Ｓ１３０６では、開発者用端末１００は、開発環境３００に対して、Ｓ１３０４で受け付けた編集操作を反映した内容でのユーザー情報の更新指示を送信する。

【０２０１】

Ｓ１３０７では、開発者用端末１００は、実行環境変更操作があったか否かを判定する。実行環境変更操作があった場合（選択中実行環境ボックス５０１に対する操作があった場合）にはＳ１３０８に進み、そうでない場合にはＳ１３０９に進む。

【０２０２】

Ｓ１３０８では、開発者用端末１００は、変更後の選択中実行環境をメモリ１０２に記録するとともに、開発環境３００に送信する。選択中実行環境が変わると、表示すべき編集対象のユーザー情報も変わるので、Ｓ１３０１に戻って再度処理を行う。

【０２０３】

Ｓ１３０９では、開発者用端末１００は、メインメニュー領域５１０に表示されたいずれかの選択肢を選択する操作である、画面切替操作があったか否かを判定する。画面切替

操作が無い場合にはＳ１３０４に戻り、画面切替操作があった場合にはユーザー情報表示処理を終了して前述した画面切替処理へ進む。

【０２０４】

図１３（ｂ）に、図１３（ａ）のユーザー情報表示処理に応答した処理である、開発環境３００におけるユーザー情報取得処理のフローチャートを示す。

【０２０５】

Ｓ１３１１では、開発環境３００は、Ｓ１３０１で開発者用端末１００から送信されたユーザー情報取得指示を受信したか否かを判定する。ユーザー情報取得指示を受信した場合はＳ１３１２に進み、そうでない場合にはＳ１３１６に進む。

【０２０６】

Ｓ１３１２では、開発環境３００は、選択中実行環境からアクセス先情報（接続先情報）を取得する。ここで取得するアクセス先情報は、選択中実行環境が図１のマルチテナント実行環境４１０であればアクセス先情報４２１であり、選択中実行環境が図１のシングルテナント実行環境であれば、アクセス先情報４５６ｂ、４６６ｂ、４７６ｂ…などのうち、選択中実行環境のものである。より詳しくは、開発環境３００は、選択中実行環境として、開発者情報３０１のうちログイン中の開発者のアカウント情報から選択中実行環境（特定環境）の実行環境ＩＤを取得してメモリ３０４に記憶している。その選択中実行環境のうち、所定のパス（実行環境ＩＤがわかれば確定するパス）に記録されたファイルにアクセスし、アクセス先情報としてそのファイルに記憶されたアクセス先のＵＲＬを取得する。所定のパスは例えば、所定の領域（フォルダ、バケット、ディレクトリ）の名称と、所定のファイル名からなる。所定の領域の名称は例えば、「固定の文字列＋実行環境ＩＤ」である。所定のファイル名は例えば、全実行環境で共通の予め定められた共通ファイル名である。この場合、所定のパスは、「固定の文字列＋実行環境ＩＤ／共通ファイル名．ｊｓｏｎ」となる。すなわち、実行環境ＩＤがわかれば確定する所定の領域にある所定のファイル（アクセス先情報ファイル）を取得し、そのファイルの中からアクセス先情報を取得し、メモリ３０４に記録する。アクセス先情報のＵＲＬは、当該実行環境中のアドレスを示しており、実行環境に保存されたユーザー情報の取得、更新を要求（リクエスト）する要求先となるＵＲＬである。このＵＲＬにリクエストしなければ、実行環境に保存されたユーザー情報の取得、更新はできない。このように、実行環境ＩＤがわからなければ、ユーザー情報の取得・更新のためのアクセス先情報は取得できない。実行環境ＩＤは、開発者が開発環境３００に認証処理を行った上でログインしないと得られない情報である。また、開発環境３００にログインできる開発者が自身のアクセス可能な実行環境ＩＤしか取得することはできない。従って開発環境３００にログインできない人物や、開発環境３００にログインできてもアクセス可能実行環境が異なる人物が不正に実行環境にアクセスしてユーザー情報が漏洩することは抑制される。すなわち、セキュリティを向上させている。

【０２０７】

Ｓ１３１３では、開発環境３００は、Ｓ１３１２で取得したアクセス先情報が示すアクセス先に、ユーザー情報取得要求（ユーザー情報を開発環境３００へ送信する処理の実行要求）と、開発者情報を送信する（送信制御）。アクセス先に送信する開発者情報は、開発者情報３０１から、暗号化されたログイン中の開発者のアカウント情報を取得する。取得するアカウント情報は、図１４（ａ）に示した開発者情報３０１の表のうち、ログイン中の開発者の１行分の情報である。すなわち、メールアドレス、パスワード、氏、名、アクセス可能な実行環境ＩＤの情報を含む。これらが暗号化され１つのデータ列となった情報を取得する。この暗号化された情報だけが漏洩したとしても、暗号化されているため、メールアドレス、パスワード、氏、名、アクセス可能な実行環境ＩＤは漏洩しない。

【０２０８】

Ｓ１３１４では、開発環境３００は、選択中実行環境から、Ｓ１３１３で送信したユーザー情報取得要求に応じたユーザー情報を受信したか否かを判定する。このユーザー情報は、後述するＳ１３２５またはＳ１３２６で実行環境から送信されるものである。ユーザ

ー情報を受信した場合はＳ１３１５に進み、そうでない場合にはＳ１３１４でユーザー情報の受信を待つ。

【０２０９】

Ｓ１３１５では、開発環境３００は、Ｓ１３１４で受信したユーザー情報を開発者用端末１００に送信する。ここで送信したユーザー情報が、前述したＳ１３０２で受信される。

【０２１０】

Ｓ１３１６では、開発環境３００は、開発者用端末１００からユーザー情報更新指示を受信したか否かを判定する。ここで受信するユーザー情報更新指示は、前述したＳ１３０６で開発者用端末１００から送信されるものである。ユーザー情報更新指示を受信した場合はＳ１３１７に進み、そうでない場合は１３１８に進む。

【０２１１】

Ｓ１３１７では、開発環境３００は、Ｓ１３１２で取得したアクセス先情報が示すアクセス先に、ユーザー情報更新要求（ユーザー情報の更新処理の実行要求）と開発者情報を送信する。アクセス先に送信する開発者情報は、Ｓ１３１３で説明したものと同様であり、開発者情報３０１から取得した、暗号化されたログイン中の開発者のアカウント情報である。

【０２１２】

Ｓ１３１８では、開発環境３００は、開発者用端末１００から選択中実行環境の変更指示を受信したか否かを判定する。この変更指示は前述したＳ１３０８で開発者用端末１００から送信されるものである。選択中実行環境の変更指示を受信した場合はＳ１３１９に進み、そうでない場合にはＳ１３１１に進んで処理を繰り返す。

【０２１３】

Ｓ１３１９では、開発環境３００は、変更後の選択中実行環境として、変更後の選択中実行環境の実行環境ＩＤをメモリ３０４に記憶する。

【０２１４】

図１３（ｃ）に、図１３（ｂ）のユーザー情報取得処理に応答した処理である、実行環境４００のうち選択中実行環境におけるユーザー情報管理処理のフローチャートを示す。この処理の動作主について、以下では、単に実行環境４００と記載するが、実際には、実行環境４００のうち選択中実行環境における実行エンジンが実行するものとする。すなわち、例えば選択中実行環境がマルチテナント実行環境４１０であれば動作主は実行エンジン４１２であり、実行エンジン４１２に含まれるプロセッサ４１３が、メモリ４１４をワークメモリとしてプログラムを実行することにより実現する。また、例えば、選択中実行環境がシングルテナント実行環境４５０であれば動作主は実行エンジン４５２であり、実行エンジン４５２に含まれるプロセッサ４５３が、メモリ４５４をワークメモリとしてプログラムを実行することにより実現する。

【０２１５】

Ｓ１３２１では、実行環境４００は、開発環境３００から送信されたユーザー情報取得要求を受信したか否かを判定する。ここで受信するユーザー情報取得要求は、前述したＳ１３１３で開発環境３００から送信されたものである。ユーザー情報取得要求を受信した場合にはＳ１３２２に進み、そうでない場合にはＳ１３２７に進む。

【０２１６】

Ｓ１３２２では、実行環境４００は、アクセス権確認処理を行う。この処理は、ユーザー情報取得要求の送信元が正当なアクセス権を有する開発者からのアクセスであるかを判定するための処理であり、この処理によって、正当なアクセス権を有する開発者以外にユーザー情報が漏洩する可能性が低減し、セキュリティが向上する。アクセス権確認処理については図１３（ｄ）を用いて後述する。

【０２１７】

Ｓ１３２３では、実行環境４００は、アクセス権確認処理の結果がエラーであるか否かを判定する。エラーであった場合には、ユーザー情報取得要求の要求元に自身の環境に保

持しているユーザー情報を送信することなく、Ｓ１３２１に進む。エラーでない場合にはＳ１３２４へ進む。

【０２１８】

Ｓ１３２４では、実行環境４００は、自身がマルチテナント実行環境４１０であるか否かを判定する。マルチテナント実行環境４１０である場合にはＳ１３２６へ進み、そうではなくシングルテナント実行環境である場合にはＳ１３２５へ進む。

【０２１９】

Ｓ１３２５では、実行環境４００は、自身の実行環境に保持しているユーザー情報を開発環境３００に送信する。シングルテナント実行環境であれば、その環境自体のオーナーが単一の開発者であるため、ユーザー情報は単一の開発者がオーナーであるアプリケーションのものしかない。従って後述するＳ１３２６のマルチテナント実行環境での処理のように、ユーザー情報のうちログイン開発者がオーナーである一部分を送信するのではなく、全体を送信する。例えば、選択中実行環境がシングルテナント実行環境４５０であれば、ユーザー情報４５１を開発環境３００に送信する。ここで送信したユーザー情報が、開発環境３００にＳ１３１４で受信され、Ｓ１３１５で開発者用端末１００に送信される。

【０２２０】

Ｓ１３２６では、実行環境４００は、自身の実行環境に保持しているユーザー情報のうち、オーナーＩＤが受信した開発者情報に一致する部分を開発環境３００に送信する。暗号化された開発者情報が開発環境３００から送信され、後述するＳ１３４１で受信され、Ｓ１３４３でデコード（暗号化解除）され、メモリ４１４に記憶されている。これが受信した開発者情報である。図１４（ｂ）に示したマルチテナント実行環境４１０のユーザー情報４１１のうち、受信した開発者情報に含まれる開発者ＩＤ（メールアドレス）と同じオーナーＩＤを持つ行の情報を取得し、開発環境３００に送信する。マルチテナント実行環境４１０のユーザー情報４１１のうち、受信した開発者情報に含まれる開発者ＩＤ（メールアドレス）と同じオーナーＩＤを持たない行の情報は開発環境３００に送信しない。マルチテナント実行環境であれば、複数の開発者がオーナーとなるユーザー情報が混在して記録されているため、ユーザー情報のうちログイン開発者がオーナーである一部分を送信する。ここで送信したユーザー情報が、開発環境３００にＳ１３１４で受信され、Ｓ１３１５で開発者用端末１００に送信される。

【０２２１】

Ｓ１３２７では、実行環境４００は、ユーザー情報更新要求を受信したか否かを判定する。ここで受信するユーザー情報更新要求は、前述したＳ１３１７で開発環境３００から送信されたものである。ユーザー情報更新要求を受信した場合にはＳ１３２８に進み、そうでない場合にはＳ１３２１に進む。

【０２２２】

Ｓ１３２８では、実行環境４００は、アクセス権確認処理を行う。この処理は、ユーザー情報更新要求の送信元が正当なアクセス権を有する開発者からのアクセスであるかを判定するための処理であり、この処理によって、正当なアクセス権を有する開発者以外からの要求によってユーザー情報が書き換えられる（更新される）可能性が低減し、セキュリティが向上する。アクセス権確認処理については図１３（ｄ）を用いて後述する。

【０２２３】

Ｓ１３２９では、実行環境４００は、アクセス権確認処理の結果がエラーであるか否かを判定する。エラーであった場合には、ユーザー情報更新要求に従った更新をすることなく、Ｓ１３２１に進む。エラーでない場合にはＳ１３３０へ進む。

【０２２４】

Ｓ１３３０では、実行環境４００は、ユーザー情報更新要求に従って自信の環境に記憶しているユーザー情報を更新する。この際、新規にユーザーを登録する場合は、マルチテナント実行環境であればログイン開発者の開発者ＩＤがオーナーＩＤとして新規ユーザーのアカウント情報に関連付けて登録される。

【０２２５】

図１３（ｄ）にアクセス権確認処理のフローチャートを示す。この処理は、前述した図１３（ｃ）のＳ１３２２、Ｓ１３２８の詳細フローチャートである。

【０２２６】

Ｓ１３４１では、実行環境４００は、開発環境３００から暗号化された開発者情報を受信したか否かを判定する。ここで受信する開発者情報は、前述したＳ１３１３またはＳ１３１７で開発環境３００から送信されたものである。暗号化された開発者情報を受信した場合にはＳ１３４３に進み、そうでない場合にはＳ１３４２に進む。

【０２２７】

Ｓ１３４２では、実行環境４００は、アクセス権確認処理の結果としてエラーである旨を記録（出力）する。この場合、前述したＳ１３２３またはＳ１３２９ではエラーと判定される。このように、ユーザー情報取得要求またはユーザー情報更新要求を受信したとしても、暗号化された開発者情報を受信しない場合にはエラーとして、要求に応じた処理は行わない（要求を許可しない）ような要求制御が行われる。従って不正なアクセスに応じた処理を抑制することができる。

【０２２８】

Ｓ１３４３では、実行環境４００は、Ｓ１３４１で受信した開発者情報を、予め保持している復号化キーを用いてデコード（暗号化解除、復号化）する。復号化された開発者情報は、実行環境から外に送信されることはないので、漏洩する可能性は低い。

【０２２９】

Ｓ１３４４では、実行環境４００は、Ｓ１３４２でデコードした開発者情報のうち、アクセス可能実行環境を示す部分に、自身の実行環境（当該実行環境）の実行環境ＩＤが含まれているか否かを判定する。含まれている場合には処理を終了し（このケースはアクセスが正当であると見なされる）、含まれていない場合にはＳ１３４２に進み、アクセス権確認処理の結果としてエラーである旨を記録（出力）する。これによって、当該実行環境に対してアクセス権のない開発者からのユーザー情報取得要求またはユーザー情報更新要求に応じた処理を行ってしまうことを防止している。従って不正なアクセスに応じた処理を抑制することができる。

【０２３０】

なお、実行環境４００は、開発環境３００以外からのアクセスは受け付けず、開発環境３００以外を送信元とするユーザー情報取得要求またはユーザー情報更新要求があっても要求に応じた処理は行わないように制御する。これによって、開発環境３００以外の装置によって暗号化された開発者情報とユーザー情報取得要求またはユーザー情報更新要求が捏造されて送信されたとしてもそれに応じた処理は行わない。従ってセキュリティが向上する。

【０２３１】

以上説明したユーザー情報表示処理によれば、実行環境４００に対する不正なアクセスによる不正な処理を抑止し、情報漏洩のリスクを低減し、セキュリティを向上させることができる。なお、上述したユーザー情報表示処理では、実行環境に保持されたユーザー情報に関しての処理を説明したが、ユーザー情報に関しての処理以外の、実行環境に関する他の処理についても同様に適用可能である。例えば。実行環境４００に保持されたデータベースやファイルに関する処理の際にも同様の処理を適用してセキュリティを向上させることができる。

【０２３２】

例えば図１２のＳ１２１０～Ｓ１２１４で説明したデータベースの表示・更新に係る処理に関して適用可能である。この場合、開発者用端末１００からのＤＢ操作指示に応じて、開発環境３００がＳ１３１２と同様に選択中実行環境からアクセス先情報ファイルを取得する。アクセス先情報ファイルには、ＤＢ操作要求を受け付けるためのリクエスト先となるＤＢ操作要求受付ＵＲＬが記録されているので、そこに対して、Ｓ１３１３と同様に、ＤＢ操作要求と開発者情報を送信する。実行環境４００は、図１３（ｃ）、図１３（ｄ）で説明した処理と同様に、ＤＢ操作要求の送信元が正当かどうかを判定する処理を行い

、正当な場合にのみ、ＤＢ操作要求に応じて自身の実行環境に保持したＤＢセットに関する処理を行う。

【０２３３】

また例えば、図１２のＳ１２１５～Ｓ１２１７で説明したファイル管理に係る処理に関して適用可能である。この場合、開発者用端末１００からのファイル管理指示に応じて、開発環境３００がＳ１３１２と同様に選択中実行環境からアクセス先情報ファイルを取得する。アクセス先情報ファイルには、ファイル管理要求を受け付けるためのリクエスト先となるファイル管理要求受付ＵＲＬが記録されているので、そこに対して、Ｓ１３１３と同様に、ファイル管理要求と開発者情報を送信する。実行環境４００は、図１３（ｃ）、図１３（ｄ）で説明した処理と同様に、ファイル管理要求の送信元が正当かどうかを判定する処理を行い、正当な場合にのみ、ファイル管理要求に応じて自身の実行環境に保持したファイルに関する処理を行う。

【０２３４】

＜ワークフロー処理＞

図１６に、ワークフロー処理のフローチャートを示す。この処理は、図１２のＳ１２０５の詳細フローチャートである。この処理は開発者用端末１００が実行する。

【０２３５】

Ｓ１６０１では、開発者用端末１００は、ディスプレイ１０５にワークフロー（ＷＦ）作成画面を表示する。図１７（ａ）に、ＷＦ作成画面の表示例を示す。サブメニュー領域５１０にはノード一覧１７１０が表示される。ノード一覧１７１０には、ノードの選択肢として、始点であるスタート１７１１、終点であるエンド１７１２、中間点であるステータス１７１３が含まれる。キャンバス５３０が表示されていた領域には、キャンバス５３０に代えてＷＦ配置領域１７００が表示される。ＷＦ配置領域１７００は、配置前は空白の領域である。ノード一覧１７１０に表示された複数の選択肢（ノードの選択肢）のいずれかを選択してＷＦ配置領域１７００に配置することができる。本実施形態では、ドラッグアンドドロップで配置するものとする。すなわち、ノード一覧１７１０に表示された複数の選択肢のいずれかをドラッグし、ＷＦ配置領域１７００にドロップすることで配置することができる。

【０２３６】

Ｓ１６０２では、開発者用端末１００は、ノードを配置する操作があったか否かを判定する。具体的には、ノード一覧１７１０に表示された複数の選択肢のいずれかドラッグし、ＷＦ配置領域１７００にドロップすることで配置する操作があったか否かを判定する。ノードを配置する操作があった場合にはＳ１６０３に進み、そうでない場合にはＳ１６０４に進む。

【０２３７】

Ｓ１６０３では、開発者用端末１００は、ノードを配置する操作に応じてＷＦ配置領域１７００に選択されたノードを配置する。

【０２３８】

Ｓ１６０４では、開発者用端末１００は、矢印を描画する操作があったか否かを判定する。矢印は、配置済みのノードとノードとを繋ぐ矢印であり、矢印の元のノードから先のノードに遷移する際の処理内容を示すものである。配置済みのノードを選択すると、選択されたノードに対して操作パスが表示され、その操作パスにマウスカーソルがある状態でドラッグを開始し（すなわち、マウスの左ボタンを押下してマウスの移動を開始し）、他のノード上でドロップする（すなわち、左ボタンの押下を放す）ことで、矢印を描画することができる。ドラッグの開始元のノードが矢印の元となり、ドロップした位置のノードが矢印の先となる。矢印を描画する操作があった場合にはＳ１６０５に進み、そうでない場合にはＳ１６０６に進む。

【０２３９】

Ｓ１６０５では、開発者用端末１００は、矢印を描画する操作に応じてＷＦ配置領域１７００に配置済みの２つのノード間に矢印オブジェクトを描画する。

【０２４０】
図１７（ｂ）に、ＷＦ配置領域１７００にノードと矢印が配置された後の表示例を示す。ＷＦ配置領域１７００には、ノード１７０１～１７０５が配置されている。ノード１７０１は始点のノードである。ノード１７０２、１７０３、１７０４は中間点（ステータス）のノードである。ノード１７０５は終点のノードである。矢印１７０６は、ノード１７０１からノード１７０２までの矢印である。矢印１７０７は、ノード１７０２からノード１７０３までの矢印である。矢印１７０８は、ノード１７０３からノード１７０４までの矢印である。矢印１７０９は、ノード１７０４からノード１７０５までの矢印である。各ノードは複数の手順（ステップ）を含むワークフローのうち１つの手順（ステップ）を示しており、矢印は手順の前後関係を示している。すなわち、図示の例では、ノード１７０１～１７０５がそれぞれ、１～５番目の手順となることが示されている。
【０２４１】
Ｓ１６０６では、開発者用端末１００は、ノードのプロパティを設定する操作があったか否かを判定する。具体的には、ＷＦ配置領域１７００に配置済みのノードを選択して右クリックをする操作があったか否かを判定する。ノードのプロパティを設定する操作があった場合にはＳ１６０７に進み、そうでない場合にはＳ１６０８に進む。
【０２４２】
Ｓ１６０７では、開発者用端末１００は、ノードに対してプロパティを設定するためのプロパティボックスを表示し、ノードのプロパティの設定操作を受け付ける。図１７（ｃ）に、ノードのプロパティボックスの表示例を示す。ノードのプロパティボックスでは、ノードのＩＤ（他のノードと重複しない初期値が設定されていて、変更可能）、ラベル（表示される名称）、生成したワークフローを配置するＵＩ画面のＩＤの設定とを受け付ける。ＵＩ画面のＩＤの設定は、ＵＩ画面のＩＤの一覧が選択肢として表示され、その選択肢の中から選択して設定することができる。
【０２４３】
Ｓ１６０８では、開発者用端末１００は、矢印のプロパティを設定する操作があったか否かを判定する。具体的には、ＷＦ配置領域１７００に描画済みの矢印を選択して右クリックをする操作があったか否かを判定する。矢印のプロパティを設定する操作があった場合にはＳ１６０９に進み、そうでない場合にはＳ１６１０に進む。
【０２４４】
Ｓ１６０９では、矢印に対してプロパティを設定するためのプロパティボックスを表示し、矢印のプロパティであるオペレーションプロパティの設定操作を受け付ける。図１７（ｅ）に、矢印のプロパティボックスの表示例を示す。矢印のプロパティボックスでは、矢印のＩＤ（他の矢印と重複しない初期値が設定されていて、変更可能）、ラベル（表示される名称）、種別（タイプ）、ユーザーの設定とを受け付ける。種別は、承認（Ｕｐｄａｔｅ）、申請（Ｃｒｅａｔ）、差戻（Ｒｅｍａｎｄ）にそれぞれ対応する選択肢のうちいずれかをを選択して設定可能である。ユーザーはＳｅｌｅｃｔＵｓｅｒとＣｕｒｒｅｎｔＵｓｅｒにそれぞれ対応する選択肢のうちいずれかを選択して設定可能である。種別は、当該矢印の元の手順から先の手順に遷移する際のアクションの種別を示している。この種別に基づいて、後述する自動的に配置されるＵＩ部品のアクションが自動設定される。
【０２４５】
Ｓ１６１０では、開発者用端末１００は、ワークフロー生成ボタン１７３０が押下されたか否かを判定する。ワークフロー生成ボタン１７３０が押下された場合はＳ１６１１に進み、そうでない場合はＳ１６１５に進む。
【０２４６】
Ｓ１６１１では、開発者用端末１００は、図１７（ｅ）に示す選択受付フォーム（ダイアログボックス）をウィザードとして表示し、作成対象のステータスの選択を受け付ける。スタート、申請中、承認済み、未申請の中かから少なくとも１つを作成対象のステータスとして選択可能である。
【０２４７】

Ｓ１６１２では、開発者用端末１００は、図１７（ｆ）に示す選択受付フォーム（ダイアログボックス）をウィザードとして表示し、タスクテーブルを追加作成対象とするかの選択を受け付ける。ここでは、タスクテーブルを生成するか否か、及び、追加する対象となるＵＩ画面のＩＤの選択が可能である。ＵＩ画面のＩＤの選択は、ＵＩ画面のＩＤの一覧が選択肢として表示され、その選択肢の中から選択することができる。

【０２４８】

Ｓ１６１３では、開発者用端末１００は、図１７（ｇ）に示す指定受付フォーム（ダイアログボックス）をウィザードとして表示し、データベースの指定（選択）を受け付ける。データベースの指定は、表示された選択肢から選択することで指定することができる。

【０２４９】

Ｓ１６１４では、開発者用端末１００は、ＷＦ配置領域１７００に配置済みのノード、矢印と、それぞれのプロパティ、及び、Ｓ１６１１～Ｓ１６１４で選択肢から選択を行って設定された内容に基づいて、ワークフロー定義を作成し、メモリ１０２の定義情報に記録する。これによって、ＵＩ画面にワークフローに基づくＵＩ部品（ＵＩコンポーネント）が自動的に配置される。自動的に配置されたＵＩ部品は自動的にワークフローに対応するアクションが設定されたものである。このように開発者ユーザーがプログラミング言語での記述をすることなく、選択肢を選択していく操作で生成したワークフローに基づいてアクションが自動的に定義されたＵＩ部品が生成される。また、図１７（ｇ）に示す指定受付フォームで指定されたデータベースに、ワークフローで用いるテーブルが作成される。また、設定内容によってＵＩ画面のキャンバスにもアクションが設定（定義）される。キャンバスに設定されたアクションは、構築されたアプリケーションにおいてそのＵＩ画面がロード（読込）された場合に行う処理である。すなわち、画面表示の際に初期動作として行う処理である。

【０２５０】

Ｓ１６１５では、開発者用端末１００は、画面切替操作があったか否かを判定する。より詳しくは、メインメニュー領域５１０に含まれるいずれかの選択肢が選択されたか否かを判定する。メインメニュー領域５１０に含まれるいずれかの選択肢が選択された場合には図１６の処理を終了し、図１２で前述した画面切替処理へ進む。そうでない場合はＳ１６０２に進む。

【０２５１】

図１８（ａ）に、図１６のＳ１６１４で追加されるデータベースのテーブルを、メインメニュー領域のデータベースボタン５１６を押下して表示させた場合の表示例を示す。すなわち、図１８（ａ）は、図１２のＳ１２１４で表示されるデータベースに関する画面の１つである。サブメニュー領域に表示されたテーブル一覧のうち、テーブル１８０１とテーブル１８０２が、ワークフローの生成に応じて自動的に追加されたテーブルである。Ｓ１６１３で指定を受け付けたデータベースに含まれる（従属する、下位階層となる）テーブルとして追加（生成）されている。

【０２５２】

図１８（ｂ）に、図１６のＳ１６１４で自動的にＵＩ部品が配置されたＵＩ画面のキャンバスの表示例を示す。キャンバス５３０に配置されたＵＩ部品１８１１～１８１３がワークフローの生成に応じて自動的に配置されたＵＩ部品である。すなわち、ＵＩ部品１８１１～１８１３は、サブメニュー領域のＵＩ部品一覧から開発者がドラッグアンドドロップを行って配置したものではない。ＵＩ部品１８１３には自動的にアクションが設定（定義）されている。ＵＩ部品１８１３は、ワークフローにおける次の手順に遷移する際のアクションが定義された自動生成部品である。また、ＵＩ部品１８１３が配置されたＵＩ画面は、ＵＩ部品１８１３に自動的に定義されたアクションの実行によるワークフローの次の手順への遷移における遷移元の画面である。

【０２５３】

図１８（ｃ）に、ボタンであるＵＩ部品１８１３を指定ＵＩ部品としてアクションボードを表示した場合の表示例を示す。すなわち、このアクションボードは、指定ＵＩ部品を

ＵＩ部品１８１３とした場合に図８で前述したＳ８０１で表示されるものである。アクションボード９１０には、開発者がプログラミング言語でのアクションの記述を行っていない状態で、初期値として図１８（ｃ）のようなアクションの文字列がプログラミング言語で記述された状態で表示される。言い換えれば、アクションボード９１０には、開発者が当該アクションに関してプログラム言語でのソースコードの記述を行っていない状態で、初期値として図１８（ｃ）のような、アクションを定義したプログラミング言語でのソースコードが表示される。図１８（ｃ）のアクションボート９１０に表示されたアクションには、図１６のワークフロー作成処理においてユーザーから受け付けた操作群に基づいて、少なくとも、データベース（３行目）、次に表示すべきＵＩ画面（６行目）が定義されている。ユーザーは、このように自動的に定義されたアクションを前述したＳ８０２～Ｓ８０８の処理によって編集可能である。すなわち、自動的に定義されたアクションの文字列をベースとして、これを一部流用し、一部改変または文字列を追加するようにプログラミング言語を入力することができる。すなわち、自動的に定義されたアクションの文字列に対する修正操作を受け付けることができる。このように、自動的に生成されて表示されたアクションのプログラミング言語をベースとすることで、開発者がゼロからプログラミング言語でアクションを記述するよりも、少ない操作手数（テキスト入力量）で簡単にアクションを設定することができる。また、自動的に生成されたアクションをそのまま使うことなく修正を加えてカスタマイズできるため、より細かい要望に応えるアクションを生成可能である。

【０２５４】

図１８（ｄ）に、ワークフローの生成に応じて自動的にアクションが設定されたＵＩ画面のキャンバスのアクションボードを表示した場合の表示例を示す。このアクションボードは、キャンバスの空白領域を指定してコンテキストメニュー処理を行ってアクションボードの表示を指示した場合に表示されるものである。後述する図１９のＳ１９０３で表示される。アクションボードには、開発者がプログラミング言語でのアクションの記述を行っていない状態で、初期値として図１８（ｄ）のようなアクションの文字列がプログラミング言語で記述された状態で表示される。言い換えれば、アクションボードには、開発者が当該アクションに関してプログラム言語でのソースコードの記述を行っていない状態で、初期値として図１８（ｄ）のような、アクションを定義したプログラミング言語でのソースコードが表示される。ユーザーは、このように自動的に定義されたアクションを後述するＳ１９０４の処理によって編集可能である。すなわち、図１８（ｃ）で説明したＵＩ部品に自動的に設定されたアクションと同様、少ない操作手数（テキスト入力量）で簡単にアクションを設定することができ、かつ、より細かい要望に応えるアクションを生成可能である。

【０２５５】

上述したワークフローの生成に応じて自動的に生成されるＵＩ部品、アクションに対して、例えば以下のようなカスタマイズを加えることが可能である。

・キャンパスのアクションボードに、ワークフロー中における今の手順（ステップ）名を表示するアクションを設定する。

・申請理由のインプットフィールド（ＵＩ部品）を配置し、そこに入力されたデータを次のステップに送るようにＵＩ部品１８１３のアクションボードに修正を加える。

・違うテーブルにデータを保存するように、部品１８１３のアクションボードに修正を加える。例えば、申請理由とケースＩＤなどを別のＤＢにも記録するようにカスタマイズする。

・キャンパスのアクションボードに、ＵＩ部品１８１２（コンボボックス）に初期入力するユーザーを、前回申請した人、というアクションをプログラミング限度で書くことにより設定する。あるいは、承認可能な人を選択肢として表示するというアクションを書く。

【０２５６】

このように、本実施形態によれば、ワークフローの基本的な部分はワークフロー作成画面での操作とウィザードへの入力内容に沿って（すなわちプログラミング言語でソースコ

ードを記述する操作を含まず、選択肢から選択する操作を含む操作群に基づいて）、簡単な操作で素早く作成することができる。なおかつ、自動的に作成されたＵＩ部品やＵＩ画面のキャンバスに設定されたアクションをプログラミング言語で修正可能に提示することで、より自由度の高い設計が可能となる。

【０２５７】

＜ＣＲＵＤ生成処理＞

図１９に、キャンバスのコンテキストメニュー処理のフローチャートを示す。この処理は、前述した図７のＳ７０３の詳細フローチャートである。この処理は開発者用端末１００が実行する。キャンバスのコンテキストメニューには、ＣＲＵＤを生成するための選択肢が含まれる。

【０２５８】

本実施形態では、ＣＲＵＤの生成の説明のための例として、ＣＲＵＤの生成前（ＣＲＵＤボタンの配置前）に、図２０（ａ）に示すような選択中ＵＩ画面のキャンバスが表示されていた場合に、キャンバスの空白領域が右クリックされてキャンバスのコンテキストメニューが表示された場合の例を説明する。ＣＲＵＤとは、ソフトウェアに要求される４つの基本機能である、データの作成（Ｃｒｅａｔｅ）、読み出し（Ｒｅａｄ）、更新（Ｕｐｄａｔｅ）、削除（Ｄｅｌｅｔｅ）の頭文字を繋げた語である。図２０（ａ）のキャンバスには、ＵＩエディタ処理において開発者によって配置されたＵＩ部品２００１～２００３が表示されている。ＵＩ部品２００１～２００３はいずれも種別がＩｎｐｕｔのＵＩ部品である。ＵＩ部品２００１はラベルとして「ＩＤ」、ＵＩ部品２００２は初期Ｖａｌｕｅとして「Ｎａｍｅ」、ＵＩ部品２００３はラベルとして「ａｇｅ」、がそれぞれ表示されている。すなわち、図２０（ａ）のキャンバスは、何らかのユーザー登録をするための画面を想定して設計されたＵＩ画面のキャンバスである。ＵＩ部品２００１は新規登録するユーザーのＩＤを、ＵＩ部品２００２は新規登録するユーザーの名前を、ＵＩの部品２００３は新規登録するユーザーの年齢を、それぞれ入力してもらうための入力欄である。ＵＩ部品２００１～２００３のＵＩ部品ＩＤはそれぞれ、「ｎｕｍｂｅｒ＿ｆｉｅｌｄ＿ＩＤ」、「ｔｅｘｔ＿ｆｉｅｌｄ＿Ｎａｍｅ」、「ｎｕｍｂｅｒ＿ｆｉｅｌｄ＿ａｇｅ」であるものとする。

【０２５９】

Ｓ１９０１では、開発者用端末１００は、マウスカーソルの近傍にキャンバスのコンテキストメニューを重畳表示する。図２０（ｂ）に、キャンバスのコンテキストメニューの表示例を示す。キャンバスのコンテキストメニューには、メニュー項目である選択肢２０１１～２０１５が表示される。

【０２６０】

Ｓ１９０２では、開発者用端末１００は、キャンバスのコンテキストメニューのうちアクションボードの表示を指示するメニュー項目（選択肢２０１３）が押下されたか否かを判定する。アクションボードの表示を指示するメニュー項目（選択肢２０１３）が押下された場合はＳ１９０３に進み、そうでない場合はＳ１９０６に進む。

【０２６１】

Ｓ１９０３では、開発者用端末１００は、メモリ１０２に保持した定義情報に基づき、キャンバスのアクションボードを表示する。キャンバスのアクションボードの表示例は例えば前述した図１８（ｄ）に示したものである。図１８（ｄ）は自動的にアクションが設定されている例である。アクションが設定されていない場合は、アクションボードは空白の状態で表示される。過去に開発者がアクションボードにアクションを入力（設定）しており、定義情報にその内容が記録されている場合は、アクションボードには設定済みのアクションを示すプログラミング言語の文字列が表示される。

【０２６２】

Ｓ１９０４では、開発者用端末１００は、アクションボードに対する開発者ユーザーからのプログラミング言語での入力操作、編集操作、及び、関数の設定操作を受け付ける（編集受付）。この処理は、前述した図８のＳ８０２～Ｓ８２３の処理と同様の処理である

ため、詳細は省略する。
【０２６３】
　Ｓ１９０５では、開発者用端末１００は、アクションボードを閉じる操作（アクションボード処理を終了する操作）があったか否かを判定する。アクションボードを閉じる操作が無い場合にはＳ１９０４に進んで処理を繰り返す。アクションボードを閉じる操作があった場合にはアクションボードを非表示とし、図１９の処理を終了して、選択中ＵＩ画面のキャンバスの表示に切り替え、ＵＩエディタ処理へ戻る。
【０２６４】
　Ｓ１９０６では、開発者用端末１００、キャンバスのコンテキストメニューのうちＣＲＵＤの生成を指示するメニュー項目（選択肢２０１５）が押下されたか否かを判定する。ＣＲＵＤの生成を指示するメニュー項目（選択肢２０１５）が押下された場合はＳ１９０７に進み、そうでない場合はＳ１９１４に進む。
【０２６５】
　Ｓ１９０７では、開発者用端末１００は、ウィザードとしてＣＲＵＤの入力フォーム１（ダイアログボックスである）を表示し、ＣＲＵＤにおいてデータを取得する対象のＵＩ部品を選択する操作を受け付ける。図２０（ｃ）に、ＣＲＵＤの入力フォーム１の表示例を示す。領域２０２１には、選択中ＵＩ画面に配置されている全ＵＩ部品のＵＩ部品ＩＤが選択肢として表示される。ユーザー（開発者）は、領域２０２１に表示された選択肢のうち、データ取得の対象としたい選択肢を選択し、選択済み領域２０２２に表示させることで選択を行う。
【０２６６】
　Ｓ１９０８では、開発者用端末１００では、入力フォーム１での選択操作が行われ、ＮＥＸＴアイコン２０２３が押下されたか否かを判定する。ＮＥＸＴアイコン２０２３が押下された場合はＳ１９０９に進み、そうでない場合はＳ１９０８でＮＥＸＴアイコン２０２３の押下を待つ。
【０２６７】
　Ｓ１９０９では、開発者用端末１００は、ＣＲＵＤの入力フォーム２（ダイアログボックスである）を表示し、ＣＲＵＤの対象となるデータベースのテーブルに用意すべきカラムの選択を受け付ける。図２０（ｄ）に、ＣＲＵＤの入力フォーム２の表示例を示す。領域２０３１には、入力フォーム１で選択されたＵＩ部品のＵＩ部品ＩＤが選択肢として表示される。ユーザー（開発者）は、領域２０３１に表示された選択肢のうち、ＣＲＵＤの対象カラムとしたい選択肢を選択し、選択済み領域２０３２に表示させることで選択を行う。
【０２６８】
　Ｓ１９１０では、開発者用端末１００は、入力フォーム２での選択操作が行われ、ＮＥＸＴアイコン２０３３が押下されたか否かを判定する。ＮＥＸＴアイコン２０３３が押下された場合はＳ１９１１に進み、そうでない場合はＳ１９１０でＮＥＸＴアイコン２０３３の押下を待つ。
【０２６９】
　Ｓ１９１１では、開発者用端末１００は、ＣＲＵＤの入力フォーム３（ダイアログボックスである）を表示し、ＣＲＵＤの対象となるデータベースとテーブルの選択操作を受け付ける。図２０（ｅ）に、ＣＲＵＤの入力フォーム３の表示例を示す。領域２０４１は、データベースの選択を受け付けるための領域である。データベースの選択は、右端のアイコンを押下して表示される選択肢から選択することができる。領域２０４２は、テーブルの名称の入力を受け付けるための領域である。
【０２７０】
　Ｓ１９１２では、開発者用端末１００は、入力フォーム３での選択／入力操作が行われ、ＦＩＮＩＳＨアイコン２０４３が押下されたか否かを判定する。ＦＩＮＩＳＨアイコン２０４３が押下された場合はＳ１９１３に進み、そうでない場合はＳ１９１２でＦＩＮＩＳＨアイコン２０４３の押下を待つ。

【０２７１】
Ｓ１９１３では、開発者用端末１００は、入力フォーム１～３に入力された内容（選択された内容）に基づいて、ＣＲＵＤボタンを生成して選択中ＵＩ画面に自動的に配置する。また、ＣＲＵＤボタンが押下されたことに応じて遷移して表示される画面であるＮｅｘｔＵＩとなるＵＩ画面を生成する。すなわち、選択中アプリにＵＩ画面が自動生成される。また、入力フォーム３に入力された内容のデータベースのテーブルを選択中実行環境のＤＢセットに自動的に生成する。
【０２７２】
Ｓ１９１４では、開発者用端末１００は、キャンバスのコンテキストメニューのうちその他のメニュー項目（選択肢２０１１、２０１２、２０１４のいずれか）が押下されたか否かを判定する。その他のメニュー項目が押下された場合はＳ１９１５に進み、そうでない場合はＳ１９１６に進む。
【０２７３】
Ｓ１９１５では、開発者用端末１００は、押下されたメニュー項目に応じた処理である、その他の処理を行う。
【０２７４】
Ｓ１９１６では、開発者用端末１００は、キャンバスのコンテキストメニューを閉じる操作（例えば、コンテキストメニューが表示された領域外をクリックする操作）があったか否かを判定する。コンテキストメニューを閉じる操作があった場合には、Ｓ１９１７でコンテキストメニューを非表示とし、図１９の処理を終了する。閉じる操作がない場合にはＳ１９０２に戻る。
【０２７５】
図２０（ｆ）に、Ｓ１９１３の処理によって自動的に配置されたＣＲＵＤボタン（自動生成部品）の表示例を示す。図２０（ｆ）は、図２０（ａ）と同じＵＩ画面のキャンバスの表示例であり、図２０（ａ）と同じＵＩ部品には同じ符号を付す。図２０（ｆ）では、図２０（ａ）に対して、ＵＩ部品２０５１～２０５３が追加されている。このＵＩ部品２０５１～２０５３がＣＲＵＤボタンである。ＣＲＵＤボタンであるＵＩ部品２０５１～２０５３には自動的にアクションが設定されている。ＣＲＵＤボタンであるＵＩ部品２０５１のアクションは、ＵＩ部品２０５１が押下されたことに応じて、入力フォーム３で設定されたデータベースのテーブルに、入力フォーム１で選択されたＵＩ部品に入力されている値を保存する（更新または追加する）処理を行い、その後、ＮｅｘｔＵＩに画面遷移する処理である。ＣＲＵＤボタンであるＵＩ部品２０５２のアクションは、ＵＩ部品２０５２が押下されたことに応じて、入力フォーム３で設定されたデータベースのテーブルのうち、入力フォーム１で選択されたＵＩ部品に入力されている値を持つデータを削除する処理を行い、その後、ＮｅｘｔＵＩに画面遷移する処理である。ＣＲＵＤボタンであるＵＩ部品２０５３のアクションは、ＵＩ部品２０５２が押下されたことに応じて、入力フォーム３で選択されたデータベースのテーブルのうち、入力フォーム１で選択されたＵＩ部品に入力されている値と一致するデータを取得し、ＮｅｘｔＵＩに画面遷移して取得したデータを表示する処理である。
【０２７６】
図２０（ｇ）に、Ｓ１９１３の処理によって自動的に生成されたＵＩ画面（ＵＩ画面ＩＤはｃｒｕｄ－ｎｅｗｕｉ－ＵＩ０１）の表示例を示す。図２０（ｇ）は、図２０（ａ）と異なるＵＩ画面のキャンバスの表示例であり、ＵＩ部品であるデータグリッド２０６１が配置されている。データグリッド２０６１は、「ＩＤ」、「ｎｕｍｂｅｒ＿ｆｉｅｌｄ＿ＩＤ」、「ｔｅｘｔ＿ｆｉｅｌｄ＿Ｎａｍｅ」、「ｎｕｍｂｅｒ＿ｆｉｅｌｄ＿ａｇｅ」の４つのカラムを有している。このうち、ＩＤ以外のカラムは、入力フォーム２で選択されたＵＩ部品に対応するものである。また、ボタンであるＵＩ部品２０６２が自動的に配置されている。このＵＩ画面（ｃｒｕｄ－ｎｅｗｕｉ－ＵＩ０１）には、キャンバスのアクションも自動的に設定されている。自動的に設定されたキャンバスのアクションは、入力フォーム３で設定されたデータベースのテーブルのうち、遷移元の画面の各種ＣＲＵ

Ｄボタンが押下された場合の、ＣＲＵＤの対象のなった行のデータをデータグリッド２０６１に表示するように制御するアクションである。このキャンバスのアクションも、前述したＳ１９０４の処理によって、自動的に設定されたアクションをベースとして開発者（ユーザー）がカスタマイズ可能である。

【０２７７】

また、自動生成されたデータグリッド２０６１も後述するグリッドのプロパティボックスやアクションボードによって各種編集が可能である。例えば、グリッドの装飾を変更したり、カラム名として表示されているラベルを「ｔｅｘｔ＿ｆｉｅｌｄ＿Ｎａｍｅ」から「名前」に変更したりすることができる。また、例えば、一部のカラムを表示しないように修正することが可能である。アプリケーションをデプロイして運用を開始した後に、何らかの事情（例えば法規制や運用方法の変更など）でカラムを増減させたい場合があり得る。例えば、将来、データグリッド２０６１のうち、「ｎｕｍｂｅｒ＿ｆｉｅｌｄ＿ａｇｅ」のカラムを表示しないように変更したい場合が発生する可能性がある。この場合、ＵＩエディタでカラムのプロパティから削除したいカラムを削除するだけで削除できる。従って改めてＣＲＵＤの入力フォーム１～３を入力しなおして生成しなおす必要がない。このように、ＣＲＵＤの入力フォーム（フィザード）への入力に応じて自動的に生成されたＵＩ部品や画面を編集できることで、アプリケーションの将来の運用の変更などにも少ない操作手数で簡単に、柔軟に対応することができる。

【０２７８】

図２１（ａ）に、Ｓ１９１３の処理によって自動的に追加されたデータベースのテーブルの例を示す。この表示例は、メインメニュー領域のデータベースボタン５１６を押下して表示させた場合の表示例である。すなわち、図２１（ａ）は、図１２のＳ１２１４で表示されるデータベースに関する画面の１つである。サブメニュー領域に表示されたデータベース「ｓａｍｐｌｅ」に含まれるテーブル一覧（ＴＡＢＬＥＳ）のうち、テーブル２１０１（ＣＲＵＤ＿Ｔ０１）が、ＣＲＵＤの生成に応じて自動的に追加されたテーブルである。テーブル２１０２は、テーブル２１０１（ＣＲＵＤ＿Ｔ０１）の詳細を表示したものであり、カラムとしてＩＤの他に、入力フォーム２で選択されたＵＩ部品のＵＩ部品ＩＤをカラム名としたカラムがあることがわかる。これらのカラムについて、データ型などのプロパティも、入力フォーム２で選択されたＵＩ部品のプロパティに応じて自動的に設定されている。

【０２７９】

図２１（ｂ）に、図１９のＳ１９１３の処理で自動的に配置されたＣＲＵＤボタンであるＵＩ部品２０５１のアクションボードを表示した場合の表示例を示す。すなわち、このアクションボードは、指定ＵＩ部品をＵＩ部品２０５１とした場合に図８で前述したＳ８０１で表示されるものである。アクションボード９１０には、開発者がプログラミング言語でのアクションの記述を行っていない状態で、初期値として図２１（ｂ）のようなアクションの文字列がプログラミング言語（ＪａｖａＳｃｒｉｐｔ）で記述された状態で表示される。言い換えれば、アクションボード９１０には、開発者が当該アクションに関してプログラム言語でのソースコードの記述を行っていない状態で、初期値として図２１（ｂ）のような、アクションを定義したプログラミング言語でのソースコードが表示される。ユーザーは、このように自動的に定義されたアクションを前述したＳ８０２～Ｓ８０８の処理によって編集可能である。すなわち、自動的に定義されたアクションの文字列をベースとして、これを一部流用し、一部改変または文字列を追加するようにプログラミング言語を入力することができる。すなわち、自動的に定義されたアクションの文字列に対する修正操作を受け付けることができる。このように、自動的に生成されて表示されたアクションのプログラミング言語をベースとすることで、開発者がゼロからプログラミング言語でアクションを記述するよりも、少ない操作手数（テキスト入力量）で簡単にアクションを設定することができる。また、自動的に生成されたアクションをそのまま使うことなく修正を加えてカスタマイズできるため、より細かい要望に応えるアクションを生成可能である。

【０２８０】
　また、図２１（ｂ）において、サブメニュー領域の関数一覧９２０には、予め用意された関数の選択肢として関数２１２１と関数２１２２が表示される。関数２１２１と関数２１２２は図１９のＳ１９１３の処理で自動的に生成された関数である。
【０２８１】
　図２１（ｃ）に、関数一覧９２０から関数２１２１を選択して関数の設定画面を表示した場合の表示例を示す。関数２１２１はＳＱＬ関数であり、ＳＱＬ関数設定画面が表示される。ＳＱＬ関数設定画面は図１０（ａ）で前述したものと同じ構成であるため、図２１（ｃ）において、図１０（ａ）と同じ設定欄／入力欄には同じ符号を付して説明を省略する。設定欄９５１には、関数名として「ＳＱＬ＿Ｓａｖｅ」が自動的に設定されている。また、設定欄９５３には、ＳＱＬ文が予め入力されている。このＳＱＬ文のうち、「ＩＮＳＥＲＴ　ＩＮＴＯ」との文字列は、データベースへの処理種別が「ＩＮＳＥＲＴ」であり、テーブルに新たに行を追加する処理であることを示している。また、「ｓａｍｐｌｅ．ＣＲＵＤ＿Ｔ０１」との文字列は、処理対象となるデータベースが「ｓａｍｐｌｅ」であり、テーブルが「ＣＲＵＤ＿Ｔ０１」であることを示している。これは、入力フォーム３に入力された内容が反映されたものである。また、それ以降の文字列は追加する行に含まれるカラムの値を示しており、入力フォーム１に入力された内容が反映されたものである。設定欄９５１、９５３の内容は、自動的に生成されたものをベースとして、前述した図８のＳ８１３～Ｓ８２０の処理によって開発者が編集（修正、削除、加筆、カスタマイズ）可能である。
【０２８２】
　図２１（ｄ）に、関数一覧９２０から関数２１２２を選択して関数の設定画面を表示した場合の表示例を示す。関数２１２２はＳＱＬ関数であり、ＳＱＬ関数設定画面が表示される。ＳＱＬ関数設定画面は図１０（ａ）で前述したものと同じ構成であるため、図２１（ｄ）において、図１０（ａ）と同じ設定欄／入力欄には同じ符号を付して説明を省略する。設定欄９５１には、関数名として「ＳＱＬ＿Ｕｐｄａｔｅ」が自動的に設定されている。また、設定欄９５３には、ＳＱＬ文が予め入力されている。このＳＱＬ文のうち、「ＵＰＤＡＴＥ」との文字列は、データベースへの処理種別が「ＵＰＤＡＴＥ」であり、テーブルの既存の行の値を更新する処理であることを示している。また、「ｓａｍｐｌｅ．ＣＲＵＤ＿Ｔ０１」との文字列は、処理対象となるデータベースが「ｓａｍｐｌｅ」であり、テーブルが「ＣＲＵＤ＿Ｔ０１」であることを示している。これは、入力フォーム３に入力された内容が反映されたものである。また、それ以降の文字列は更新するカラムの値を示しており、入力フォーム１に入力された内容が反映されたものである。設定欄９５１、９５３の内容は、自動的に生成されたものをベースとして、前述した図８のＳ８１３～Ｓ８２０の処理によって開発者が編集（修正、削除、加筆、カスタマイズ）可能である。
【０２８３】
　図２１（ｂ）のアクションボード９１０に図示した、自動的に生成されるアクションの内容について詳細に説明する。図２１（ｂ）のアクションボード９１０に記述されたＪａｖａＳｃｒｉｐｔの文字列の各行は、下記の意味である。
１行目…ｐａｒａｍｓという変数名で、２～５行目の値を含む値のセットを定義する。
２行目…ＩＤとなる数字の定義。通し番号となる。
３～５行目…入力フォーム１で選択されたＵＩ部品に入力されている値を取得する。なお、値を取得するという意味で、「＝」の右側に「＄ｕｉ．ＵＩ部品ＩＤ．対象となる情報の種別」が記載されているが、この記述方式自体はＪａｖａＳｃｒｉｐｔの標準の記載方法ではなく、本システム独自のものである。これについて詳細は後述する。
６行目…２～５行目で定義したｐａｒａｍｓを引数として関数ＳＱＬＵｐｄａｔｅを実行する。すなわち、入力フォーム１で選択されたＵＩ部品に入力されている値で、入力フォーム３で設定したデータベースのテーブルを更新する。
７行目…Ｕｐｄａｔｅによって影響のある行（Ｒｏｗ）を取得する。

８行目…７行目で取得できた行があるか判定する。すなわち、Ｕｐｄａｔｅできた行があるか判定する。
９行目…８行目が真であれば、「Ｓｕｃｃｅｓｓｆｕｌｌｙ　Ｕｐｄａｔｅｄ」（更新できました）と表示する。
１０行目…ＮｅｘｔＵＩとしてＵＩ画面ＩＤが「ｃｒｕｄ－ｎｅｗｕｉ－ＵＩ０１」であるＵＩ画面へ遷移する。
１２行目…７行目で取得できた行が０であるか判定する。これが真となる場合には、今回取得したｐａｒａｍｓに対応する既存の行が無かったことを意味するため、新しい行を追加して新規登録することになる。
１３行目…１２行目が真である場合に、２～５行目で定義したｐａｒａｍｓを引数として関数ＳＱＬＳａｖｅを実行する。すなわち、入力フォーム１で選択されたＵＩ部品に入力されている値で、入力フォーム３で設定したデータベースのテーブルに行を追加して新規保存を行う。
１４行目…１３行目での新規保存で影響のあった行（すなわち、追加した行）を取得する。
１５行目…１４行目で取得できた行があるかを判定する。すなわち、行を追加しての新規登録が成功したかを判定する。
１６行目…１５行目が真であれば、「Ｓｕｃｃｅｓｓｆｕｌｌｙ　Ｉｎｓｅｒｔｅｄ」（追加が成功した）と表示する。
１７行目…ＮｅｘｔＵＩとしてＵＩ画面ＩＤが「ｃｒｕｄ－ｎｅｗｕｉ－ＵＩ０１」であるＵＩ画面へ遷移する。
１９、２０行目…８行目も１２行目も偽である場合は、「Ｉｎｓｅｒｔｉｏｎ　Ｆａｉｌｅｄ」（失敗した）と表示する。

【０２８４】

このように自動的に生成され、開発者（ユーザー）が修正できるようにプログラミング言語でアクションボードに提示された内容に対して、開発者（ユーザー）は、例えば、下記のようなカスタマイズを行うことができる。
・メッセージの内容（アクションの実行に応じて表示されるメッセージ）を変更したいので、９行目、１６行目、２０行目のメッセージ内容だけ書き換える。
・ＳＡＶＥボタンの押下に応じて表示する画面（アクションの実行に応じて表示される画面）は自分で作ったものにしたいので、１０行目、１７行目のＮｅｘｔＵＩのＵＩ画面ＩＤの部分（ｃｒｕｄ－ｎｅｗｕｉ－ＵＩ０１）だけ書き換える。
・年齢を登録しない運用に変更したいので５行目を削除する。すなわち、アクションの実行の際に情報を取得及び／または出力する対象となるコンポーネントを変更する。
・複数のデータベースに登録したいので、ＳＱＬ関数（創作関数）を新たに作成（追加）し、書き加える。ＳＱＬ関数を新たに作成する際には、設定欄９５３には、自動的に作成された「ＳＱＬＳａｖｅ」や「ＳＱＬＵｐｄａｔｅ」のＳＱＬ文をコピー＆ペーストし、対象となるデータベースとテーブルの部分だけ修正する。

【０２８５】

このように、プログラミング言語でのアクションの記述が全くない状態から開発者が所望するＣＲＵＤボタンのアクションを作成するよりは、自動的に生成されたものからカスタマイズする方が、大きく操作手数を削減することができる。また、自由にカスタマイズできるので、作成できるアクションの内容の自由度も大きい。本実施形態によれば、開発者（ユーザー）がプログラム言語での記述をすることなく定義されたアクション（入力フォーム１～３へ入力する操作群に基づいて定義されたアクション）を、より柔軟に修正して利用できる。

【０２８６】

＜開発者アカウント登録処理＞

図２２（ａ）に、開発者用端末１００における開発者アカウント登録処理のフローチャートを示す。この処理は、前述した図３のＳ３０６の詳細フローチャートである。

【０２８７】
Ｓ２２０１では、開発者用端末１００は、新規に登録するアカウント情報として、メールアドレス、氏、名の入力を受け付ける入力欄をディスプレイ１０５に表示し、ユーザー（開発者）からの各入力欄への入力を受け付ける。
【０２８８】
Ｓ２２０２では、開発者用端末１００は、Ｓ２２０１で入力されたアカウント情報と、当該アカウント情報を新規に登録するリクエストである登録要求とを、開発環境３００に送信する。
【０２８９】
Ｓ２２０３では、開発者用端末１００は、ワンタイムパスワードの入力を受け付ける受付画面をディスプレイ１０５に表示し、ユーザー（開発者）からのワンタイムパスワードの入力を受け付ける。開発環境３００にアカウント情報の登録依頼を送信すると、開発環境３００から、アカウント情報に含まれるメールアドレスに対し、当該メールアドレスが有効であることを確認するためにワンタイムパスワードが送られてくる。開発者は、そのメールアドレスの所有者であるため、そのメールアドレスに送られてきたワンタイムパスワードを見て、ディスプレイ１０５に表示されたワンタイムパスワードの受付画面へ入力する。
【０２９０】
Ｓ２２０４では、開発者用端末１００は、Ｓ２２０３で入力されたワンタイムパスワードを開発環境３００に送信する。
【０２９１】
Ｓ２２０５では、開発者用端末１００は、開発環境３００からログインＯＫ通知を受信したか否かを判定する。ログインＯＫ通知を受信した場合はＳ２２０９に進み、そうでない場合はＳ２２０６に進む。
【０２９２】
Ｓ２２０６では、開発者用端末１００は、ワンタイムパスワードの再送を開発環境３００に要求するための再送指示画面を表示する。
【０２９３】
Ｓ２２０７では、開発者用端末１００は、開発者（ユーザー）からのワンタイムパスワードの再送指示操作があったか否かを判定する。再送指示操作があった場合にＳ２２０８に進む。再送指示操作がない場合に開発者用端末１００は再送指示操作があるまで待機する。
【０２９４】
Ｓ２２０８では、開発者用端末１００は、開発環境３００に、ワンタイムパスワードの再送要求を送信する。
【０２９５】
Ｓ２２０９では、開発者用端末１００は、パスワードの登録画面を開発者用端末１００に送信し、パスワードの登録（パスワードとしたいデータの入力）を受け付ける。
【０２９６】
Ｓ２２１０では、開発者用端末１００では、Ｓ２２０９で受けつけたパスワードとしたいデータを開発環境３００に送信する。
【０２９７】
図２２（ｂ）に、開発環境３００における開発者アカウント登録処理のフローチャートを示す。この処理は、図２２（ａ）の開発者用端末１００における開発者アカウント登録処理と連動する開発環境３００側の処理である。
【０２９８】
Ｓ２２２１では、開発環境３００は、アカウント情報と、当該アカウント情報を新規に登録するリクエストである登録要求（登録指示）を開発者用端末１００から受信したか否かを判定する。このアカウント情報と登録要求は図２２（ａ）のＳ２２０２で開発者用端末１００から送信されたものである。アカウント情報と、当該アカウント情報を新規に登

録するリクエストである登録要求を開発者用端末１００から受信した場合はＳ２２２２に進み、そうでない場合は受信するまで待機する。

【０２９９】

Ｓ２２２２では、開発環境３００は、開発者情報３０１に行を追加し、Ｓ２２２１で受信したアカウント情報を記録する。また、この行のカラム「Ｖｅｒｉｆｙ」の値は「ｕｎｃｏｎｆｉｒｍｅｄ」を記録する。開発者情報３０１にあるカラムは、図１４（ａ）に示したものである。開発者情報３０１の１行分が１人分のアカウント情報であり、同じ行に記録された各カラム（列）の情報は互いに同じ開発者のアカウント情報として関連付けて記録されたものである。

【０３００】

Ｓ２２２３では、開発環境３００は、外部のワンタイムパスワード発行システム（不図示）に対して、Ｓ２２２１で受信したアカウント情報に含まれるメールアドレスを送信先としてワンタイムパスワードの発行と送信をするように指示を行う。これによって、外部のワンタイムパスワード発行システムから、開発者のメールアドレス宛にワンタイムパスワードが送信される。なお、本実施形態では外部のワンタイムパスワード発行システムにワンタイムパスワードの送信を指示するものとしたが、開発環境３００自体がワンタイムパスワードを送信するものとしても良い。

【０３０１】

Ｓ２２２４では、開発環境３００は、開発者用端末１００からワンタイムパスワードを受信したか否かを判定する。ここで受信するワンタイムパスワードは、図２２（ａ）のＳ２２０４で開発者用端末１００から送信されたものである。ワンタイムパスワードを受信した場合にはＳ２２２５に進み、そうでない場合にはＳ２２２４で待機する。

【０３０２】

Ｓ２２２５では、開発環境３００は、外部のワンタイムパスワード発行システムに対して、Ｓ２２２５で受信したワンタイムパスワードを送信し、発行されたワンタイムパスワードと整合するかどうかの確認指示を送信する。すなわち、ワンタイムパスワードの照合を指示する。

【０３０３】

Ｓ２２２６では、開発環境３００は、外部のワンタイムパスワード発行システムから、正しいワンタイムパスワードであった旨（照合ＯＫであった旨）の通知を受信したか否かを判定する。正であった場合（照合ＯＫであった場合）にはＳ２２２８に進み、そうでない場合にはＳ２２７へ進む。

【０３０４】

Ｓ２２２７では、開発環境３００は、開発者用端末１００から、ワンタイムパスワードの再送要求を受信したか否かを判定する。ここで受信する要求は、図２２（ａ）のＳ２２０８で開発者用端末１００から送信されたものである。ワンタイムパスワードの再送要求を受信した場合はＳ２２２３へ進み、そうでない場合はＳ２２２７で待機する。

【０３０５】

Ｓ２２２８では、開発環境３００は、Ｓ２２２２で開発者情報３０１に追加したアカウントを有効化する。具体的には、Ｓ２２２２で開発者情報３０１に追加したアカウント情報のうち、カラム「Ｖｅｒｉｆｙ」の値を「ｃｏｎｆｉｒｍｅｄ」に変更する。カラム「Ｖｅｒｉｆｙ」の値が「ｃｏｎｆｉｒｍｅｄ」であるアカウント情報は、登録されたメールアドレスがワンタイムパスワードの照合によって有効であることが確認されたものであり、登録済みの有効なアカウントとして見なされる。なお、本実施形態ではメールアドレスによる認証を行うが、アカウント情報として電話番号やＳＮＳアカウントなどの開発者への他の連絡先が登録されるようにした場合には、他の連絡先を用いた認証としてもよい。

【０３０６】

Ｓ２２２９では、開発環境３００は、開発者情報３０１に登録された有効なアカウント数（「Ｖｅｒｉｆｙ」の値が「ｃｏｎｆｉｒｍｅｄ」であるアカウントの数）が予め定め

られた閾値２（例えば９０００）より大きいかどうかを判定する。有効なアカウント数が閾値２より大きければＳ２２３０に進み、有効なアカウント数が閾値２以下であればＳ２２３２に進む。

【０３０７】

Ｓ２２３０は、開発環境３００は、マルチテナント実行環境４１０にＤＢインスタンスを追加済みであるか否かを判定する。ＤＢインスタンスを追加済みであればＳ２２３２に進み、追加済みでなければＳ２２３１に進む。

【０３０８】

Ｓ２２３１では、開発環境３００は、マルチテナント実行環境４１０のＤＢセット４３０に、新たなＤＢインスタンスを追加する。ＤＢインスタンスの新規追加（新規作成）には数十分の処理時間を要するため、有効アカウント数が後述する閾値１（＞閾値２）を超えて接続先のＤＢインスタンスを変更する前に、閾値１を超えた時点で事前に準備しておく。このようにすることで、接続先のＤＢインスタンスの変更があった際に、遅延なく、すぐに新しいＤＢインスタンスに接続できる。

【０３０９】

Ｓ２２３２では、開発環境３００は、開発者情報３０１に登録された有効なアカウント数（「Ｖｅｒｉｆｙ」の値が「ｃｏｎｆｉｒｍｅｄ」であるアカウントの数）が予め定められた閾値１（閾値２より大きい値で、例えば開発者用端末１００００）より大きいかどうかを判定する。有効なアカウント数が閾値１より大きければＳ２２３３に進み、有効なアカウント数が閾値１以下であればＳ２２３６に進む。

【０３１０】

Ｓ２２３３では、開発環境３００は、新たに追加されるアカウント情報に割り当てるマルチテナント実行環境のＤＢインスタンス名としてメモリ３０４に保持していたＤＢインスタンス名を、Ｓ２２３１で追加しておいた新たなＤＢインスタンスのＤＢインスタンス名に更新済であるか否かを判定する。更新済みである場合にはＳ２２３６へ進み、そうでない場合にはＳ２２３４へ進む。

【０３１１】

Ｓ２２３４では、開発環境３００は、新たに追加されるアカウント情報に割り当てるマルチテナント実行環境のＤＢインスタンス名としてメモリ３０４に保持していたＤＢインスタンス名を、Ｓ２２３１で追加しておいた新たなＤＢインスタンスのＤＢインスタンス名に更新する。ＤＢインスタンス名は、データベース環境の識別子である。これによって、例えば、これ以前に追加されたアカウントには「ＤＢインスタンス名１」を割当てていたのが、これ以降に追加されるアカウントには「ＤＢインスタンス名２」が割り当てられるようになる。１つのＤＢインスタンス（データベース環境）に接続するアカウント数が多くなると、１度にデータベースにアクセスされる数が増加し、アクセスが集中した場合にレスポンスが低下するなどのパフォーマンスの低下を招くことがある。これに対し、Ｓ２２３３の処理によって、アカウント数が閾値１を超えた場合には、それ以降に登録されたアカウントは、それ以前とは異なる新たなＤＢインスタンス（データベース環境）へ接続するように設定されるため、１つのＤＢインスタンスを使用する開発者の数が過大となりアクセスが集中してしまう可能性を低減することができる。従って、データベースへのアクセスに関してレスポンスが低下するなどのパフォーマンスが低下する可能性を低減することができる。

【０３１２】

Ｓ２２３５では、開発環境３００は、閾値１＝閾値１＋閾値１に更新する。また、閾値２＝閾値２＋閾値１に更新する。これによって、例えば、更新前の閾値１が開発者用端末１００００、更新前の閾値２が９０００であった場合には、閾値１＝２００００、閾値２＝１９０００となる。すなわち、更に開発者用端末１００００アカウント増える度（閾値１の数だけ有効なアカウント数が増える度）に、Ｓ２２３１の処理とＳ２２３４の処理が行われる。

【０３１３】

Ｓ２２３６では、開発環境３００は、Ｓ２２２２で追加されたアカウント情報（今回追加されたアカウント）の接続先ＤＢインスタンスとしてメモリ３０４に保持されたＤＢインスタンス名を割当てる。より具体的には、開発者情報３０１のうち、今回追加されたアカウントの行の、接続先ＤＢインスタンス名を示すカラム（図１４（ａ）の一番右の列）に、接続先ＤＢインスタンスとしてメモリ３０４に保持されたＤＢインスタンス名を記録する。Ｓ２２３２～Ｓ２２３５で説明した処理によって、例えば、閾値１が開発者用端末１００００で、今回追加されたアカウントが９９９９個目の有効なアカウントであれば「ＤＢインスタンス名１」が記録され。今回追加されたアカウントが開発者用端末１０００１個目の有効なアカウントであれば「ＤＢインスタンス名２」が記録される。

【０３１４】

Ｓ２２３７では、開発環境３００は、開発者用端末１００に、ログインＯＫ通知を送信する。ここで送信したログインＯＫ通知が、前述したＳ２２０５において開発者用端末１００によって受信される。

【０３１５】

Ｓ２２３８では、開発環境３００は、前述したＳ２２１０で開発者用端末１００から送信されたパスワードとしたいデータを受信し、今回追加したアカウント情報のパスワードとして登録する。より詳しくは、開発者情報３０１のうち、今回追加されたアカウントの行の、パスワードのカラムに記録する。

【０３１６】

図２３（ａ）に、マルチテナント実行環境４１０にあるＤＢセット４３０の詳細を示す。ＤＢセット４３０には、最初は１つまたは複数である固定数のＤＢ環境（ＤＢインスタンス）が用意されている。本実施形態では固定数＝１、つまり最初はＤＢインスタンス１（２３１０）だけが用意されているものとする。ＤＢインスタンス１の中には、開発者情報ＤＢ２３１１と、開発者毎のＤＢが含まれる。開発者情報ＤＢ２３１１には、各開発者（各アカウント）に用意されるデータベースがどれであるかを示している。例えば、開発者ＡのＤＢ情報２３１２は、開発者ＡのＤＢがＤＢ１である、という対応付けを示す情報である。開発者毎のＤＢにはＤＢ１（２３１４）、ＤＢ２（２３１５）、ＤＢ３（２３１６）…とあり、開発者情報３０１に有効なアカウントが登録されるたびに増える。ただし上限は閾値１と同じ数までであり、閾値１が開発者用端末１００００の場合、ＤＢｍａｘ（２３１７）が開発者用端末１００００個目のＤＢであり、それ以降は別のＤＢインスタンスにＤＢが作られる。発明者毎のＤＢの中にはさらに１以上のテーブルが含まれる。例えばＤＢ１（２３１４）にはテーブル１、テーブル２、…と、複数のテーブルが含まれる。なお、各ＤＢインスタンス（データベース環境）は、書き込み可能なデータベースインスタンスである。

【０３１７】

Ｓ２２３１の処理によって、ＤＢインスタンス１のＤＢの数が上限（閾値１）に達する前に、次のＤＢインスタンス２が生成される。そして、ＤＢインスタンス１にＤＢｍａｘ（２３１７）まで生成された状態から、更に有効な開発者アカウントが開発者情報３０１に追加登録された場合、Ｓ２２３２でＹｅｓとなり、追加登録された開発者はＤＢインスタンス２（２３２０）に接続するように開発者情報３０１に登録される。そして、ＤＢインスタンス２（２３２０）にそれ以降追加されたアカウント（開発者）用のＤＢが生成されていく。なお、ＤＢインスタンス２（２３２０）に入るＤＢの数も閾値１まで（通算、閾値１×２まで）であり、それ以降に登録されたアカウントは、同様にして次のＤＢインスタンスに接続するように制御される。

【０３１８】

図２３（ｂ）に、シングルテナント実行環境のＤＢセット４５７の詳細を示す。シングルテナント実行環境は１つのアカウント（１人の開発者）に専用であるため、ＤＢセット４５７も１つのアカウント（１人の開発者）に専用である。そのため、レスポンスが低下するほどのＤＢの増加、アクセスの集中が起こる可能性は低い。従って、ＤＢセット４５７には、ＤＢ環境としてはＤＢインスタンス１（２３５０）の１つだけが用意される。ま

た、開発者情報ＤＢ２３５１には１アカウント文の情報（例えば開発者ＡのＤＢ情報２３５２）だけが記録される。格納されるＤＢの数にも上限となる閾値は用意されない。また、図２２（ｂ）で説明したように、登録されるアカウント情報に関連付けて、接続先となるシングルテナント実行環境のデータベース環境を決定する制御は、開発者の新規アカウントの登録（追加）の際には行わない。

【０３１９】

＜デプロイ処理＞

図２４（ａ）に、開発者用端末１００におけるデプロイ処理のフローチャートを示す。この処理は、開発者用端末１００が実行する処理であり、図４のＳ４２６の詳細フローチャートである。

【０３２０】

図２５（ａ）に、選択中アプリがモバイル用アプリである場合のＵＩエディタ画面の表示例を示す。キャンバス２５０１はモバイル用の形状のキャンバスであり、スマートフォンを模した形状としている。デプロイボタン５０６が押下されると図２４（ａ）の処理が開始される。

【０３２１】

Ｓ２４０１では、開発者用端末１００は、開発環境３００に対して、選択中アプリのデプロイを指示する情報を送信する。なお、デプロイ指示を行う前に開発者用端末１００のメモリ１０２に保持された最新の定義情報を開発環境３００に保存する保存処理を行ってからデプロイ指示を行うようにしてもよい。

【０３２２】

Ｓ２４０２では、開発者用端末１００は、処理待ち画面を表示する。処理待ち画面では、デプロイの処理中であることを示すガイド情報が表示される。

【０３２３】

Ｓ２４０３では、開発者用端末１００は、開発環境３００からデプロイの失敗通知を受信したか否かを判定する。失敗通知を受信した場合にはＳ２４０４に進み、そうでない場合にはＳ２４０５へ進む。

【０３２４】

Ｓ２４０４では、開発者用端末１００は、デプロイが失敗した旨のエラー表示を行う。デプロイが失敗した場合は、後述のＳ２４０７、Ｓ２４０９の処理は行われない。

【０３２５】

Ｓ２４０５では、開発者用端末１００は、開発環境３００からデプロイの成功通知を受信したか否かを判定する。成功通知には、デプロイ済のアプリにアクセスするためのアクセス先の情報（ＵＲＬ）が含まれる。成功通知を受信した場合にはＳ２４０６に進み、そうでない場合にはＳ２４０３へ進む。

【０３２６】

Ｓ２４０６では、開発者用端末１００は、選択中アプリ（デプロイしたアプリ）がモバイル用であるか否かを判定する。モバイル用である場合にはＳ２４０８に進み、そうでない場合（すなわちデスクトップ用である場合）にはＳ２４０７に進む。モバイル用でない場合には後述するＳ２４０９のＱＲコード（登録商標）の表示は行わない。

【０３２７】

Ｓ２４０７では、開発者用端末１００は、デプロイ指示を受け付ける前にブラウザソフトで表示していた画面（例えばキャンバスを含むＵＩエディタ画面）とは異なるブラウザソフトの新しいタブで、成功通知に含まれるデプロイ済みのアプリのＵＲＬ（実行環境にアクセスするＵＲＬ）にアクセスした画面を表示する。これによって、開発者は、デプロイが成功したことをすぐに認識することができる。また、新しいタブに、デプロイ済みアプリの認証画面またはイニシャルＵＩが表示され、新しいタブに対して操作することで、デプロイ済み（実行環境に構築済み）の実際のアプリの表示内容や動作をすぐさま確認・検証することができる。このＳ２４０７の処理によって、デプロイ済みのアプリのＵＲＬ（実行環境にアクセスするＵＲＬ）にアクセスして行うアプリの実行処理が行われる。ア

プリの実行処理の詳細については図３３（ｃ）または図３５（ｃ）のフローチャートを用いて説明する。

【０３２８】

Ｓ２４０８では、開発者用端末１００は、成功通知に含まれるアプリのＵＲＬをＱＲコード化する処理を行う。すなわち、アプリのＵＲＬの情報を含むＱＲコード（２次元コード）を生成する。

【０３２９】

Ｓ２４０９では、開発者用端末１００は、Ｓ２４０９で生成したＱＲコードを表示する。ＱＲコードはデプロイの完了（成功）に応じて表示されるため、開発者は、デプロイが成功したタイミングを認識することができる。図２５（ｂ）に、Ｓ２４０９でのＱＲコードの表示例を示す。図２５（ｂ）は、図２５（ａ）の状態からデプロイボタン５０６が押下されてデプロイ指示がなされ、デプロイが成功した場合に遷移する画面の表示例である。モバイル用のキャンバスに重畳して、ＱＲコードダイアログ２５１０が表示される。ＱＲコードダイアログ２５１０には、Ｓ２４０８で生成したＱＲコード２５１１と、閉じるボタン２５１２と、新しいタブで開くボタン２５１３が表示される。なお、ＱＲコード２５１１が有する情報であるアプリのＵＲＬは、ＵＩエディタ画面を表示しているブラウザのアドレスバーに表示されたＵＲＬ２５０４（本実施形態の開発システム（アプリケーション開発プラットフォーム）のＵＲＬであって、開発者用端末１００にアクセスするＵＲＬ）とは別のＵＲＬである。ＱＲコード２５１１は、デプロイが完了したことによって表示されるものであり、デプロイされたアプリを実行するためのアドレスであって、デプロイがされた場所（実行環境）を示すアクセス先情報である。

【０３３０】

開発者（ユーザー）は、開発者用端末１００のディスプレイ１０５に表示されたＱＲコード２５１１を、手持ちのスマートフォン（モバイル端末）で撮影して読み取ることで、手持ちのスマートフォンにおいて、デプロイ済みのアプリのＵＲＬ（実行環境にアクセスするＵＲＬ）にアクセスした画面を表示させることができる。従って少ない操作手数で容易にデプロイ済みのＵＲＬにアクセスすることができる。この場合の手持ちのスマートフォンは、アプリユーザー用端末２０１として機能する。これによって、手持ちのスマートフォンに、デプロイ済みアプリの認証画面またはイニシャルＵＩが表示され、手持ちのスマートフォンに対して操作することで、デプロイ済み（実行環境に構築済み）の実際のアプリの表示内容や動作をすぐさま確認・検証することができる。モバイル用アプリの動作を、開発者用端末１００ではなくモバイル端末で確認・検証できるため、開発者用端末１００で確認・検証するよりも、モバイル用アプリとして好適であるかを確認しやすい。なお、ＱＲコード２５１１に加えて、あるいは代えて、成功通知に含まれるアプリのＵＲＬを読み取れるコード画像として、ＱＲコード以外のコード画像を表示するようにしてもよい。例えば、バーコード、カメレオンコード（登録商標）などでもよい。また、ＱＲコードダイアログ２５１０に、成功通知に含まれるアプリのＵＲＬ自体を文字列で表示するようにしてもよい。

【０３３１】

Ｓ２４１０では、開発者用端末１００は、新しいタブで開くボタン２５１３が押下されたか否かを判定する。新しいタブで開くボタン２５１３が押下された場合はＳ２４０７に進み、そうでない場合にはＳ２４１１に進む。

【０３３２】

Ｓ２４１１では、開発者用端末１００は、閉じるボタン２５１２が押下されたか否かを判定する。閉じるボタン２５１２が押下された場合はＳ２４１２に進み、そうでない場合にはＳ２４１０に進む、

Ｓ２４１２では、開発者用端末１００は、ＱＲコードダイアログ２５１０を非表示とし、デプロイの指示の前に表示していた画面に戻る。

【０３３３】

図２４（ｂ）に、デプロイ処理のフローチャートを示す。この処理は、開発環境３００

が実行する処理であり、図２４（ａ）のＳ２４０１で開発者用端末１００から送信されたデプロイ指示を開発環境３００が受信すると開始される処理である。

【０３３４】

Ｓ２４２１では、開発環境３００は、開発環境３００のストレージ３２０のうち、ログインしている開発者の開発者領域に保存されたアプリの定義情報のうち、デプロイ対象となる選択中アプリの定義情報を取得する。ここで取得する定義情報には、後述するＵＩ定義情報（ｕｉＤｅｆ．ｊｓｏｎ）と、実行環境用プログラム（ＪａｖａＳｃｒｉｐｔ）が含まれる。

【０３３５】

Ｓ２４２２では、開発環境３００は、選択中実行環境がマルチテナント実行環境であるか否かを判定する。マルチテナント実行環境である場合にはＳ２４２３に進み、そうでない場合（シングルテナント実行環境である場合）にはＳ２４２４に進む。

【０３３６】

Ｓ２４２３では、開発環境３００は、マルチテナント実行環境４１０のストレージ４２０のうち、ログインしている開発者の開発者領域（専用フォルダ）に、Ｓ２４２１で取得した選択中アプリの定義情報を記憶させる。より具体的には、Ｓ２４２１で取得した選択中アプリの定義情報をマルチテナント実行環境４１０に送信し、ログインしている開発者の開発者領域（開発者毎のフォルダ）に記録するようにマルチテナント実行環境４１０に指示する。この記録が問題なく完了するとデプロイ成功となる。

【０３３７】

Ｓ２４２４では、開発環境３００は、選択中実行環境であるシングルナント実行環境のストレージ（シングルテナント実行環境のバケットの直下のフォルダ）に、Ｓ２４２１で取得した選択中アプリの定義情報を記憶させる。より具体的には、Ｓ２４２１で取得した選択中アプリの定義情報を選択中実行環境であるシングルナント実行環境に送信し、バケットの直下のフォルダに記録するようにマルチテナント実行環境４１０に指示する。この記録が問題なく完了するとデプロイ成功となる。

【０３３８】

Ｓ２４２５では、開発環境３００は、デプロイが失敗したか否かを判定する。デプロイが失敗した場合にはＳ２４２６に進み、そうでない場合にはＳ２４２７に進む。通信障害などで定義情報を実行環境に送信・記録できなかった場合にはデプロイ失敗となることがある。

【０３３９】

Ｓ２４２６では、開発環境３００は、デプロイが失敗した旨を示す失敗通知を開発者用端末１００に送信する。

【０３４０】

Ｓ２４２７では、開発環境３００は、デプロイが完了（成功）したか否かを判定する。デプロイが完了した場合にはＳ２４２８に進み、そうでない場合にはＳ２４２５に進む。

【０３４１】

Ｓ２４２８では、開発環境３００は、デプロイ済みのアプリのＵＲＬの情報（実行環境にデプロイ済のアプリにアクセスするためのアクセス先の情報）を含むデプロイの成功通知を開発者用端末１００に送信する。ここで送信した成功通知が、前述の図２４（ａ）のＳ２４０５で開発者用端末１００に受信される。

【０３４２】

＜ＵＩ画面のテンプレート＞

ＵＩ画面のテンプレートについて説明する。ＵＩ画面には、ユーザーが作成したＵＩ画面とは別に、テンプレ―トとして予め用意されているＵＩ画面（テンプレート画面）がある。テンプレート画面は、ユーザーがそのＵＩ画面に過去にＵＩ部品を配置していなくても、予め定められたアクションが定義された少なくとも１つの雛形コンポーネントが予め決まった位置に配置された画面である。テンプレート画面は、予め用意された情報に対してカスタマイズが可能である。より詳しくは、ＵＩエディタでキャンバスにテンプレート

画面を表示させ、図４で説明したＵＩエディタ処理によって編集が可能である。
【０３４３】
図２６（ａ）に、サブメニュー領域５２０にＵＩ画面一覧を選択肢として表示した場合の、テンプレート画面の選択肢の表示例を示す。この表示は、前述した図４のＳ４０１の処理によって行われるものである。図２６（ａ）は、複数の種別に分類されたＵＩ画面のうち、ＡｕｔｈｅｎｔｉｃａｔｉｏｎＵＩ（認証用のＵＩ画面、認証用画面）のグループに分類されるＵＩ画面の選択肢の一覧を展開して表示した例である。グループ名２６１０は展開して表示しているグループがＡｕｔｈｅｎｔｉｃａｔｉｏｎＵＩであることを示している。グループ名２６１０が押下されるとＡｕｔｈｅｎｔｉｃａｔｉｏｎＵＩの選択肢の展開を折り畳み、他のグループのグループ名がサブメニュー領域５２０に表示される。選択肢２６１１～２６２２がそれぞれ１つのＵＩ画面の選択肢であり、いずれもテンプレート画面として予め用意されたものである。
【０３４４】
図２６（ｂ）に、選択肢２６１１を選択して、選択肢２６１１に対応するテンプレート画面（ＵＩ画面）である「Ｓｉｇｎ　ｉｎ　ＩＤ」をキャンバス２６３０に表示した場合の表示例を示す。図２６（ｂ）の画面は、選択肢２６１１が押下されたことに応じて、前述した図４のＳ４０４で表示される。キャンバス２６３０には、雛形部品２６３１～２６３４が表示されている。雛形部品２６３１～２６３４は、ユーザー（開発者）がサブメニュー領域５２０のＵＩ部品一覧からドラッグ＆ドロップで配置したものではなく、予め表示位置（配置）や色、サイズ、表示内容が定義された雛形のＵＩ部品である。雛形部品２６３１～２６３４はそれぞれ以下のようなＵＩ部品である。
・雛形部品２６３１：Ｏｕｔｐｕｔ　Ｆｉｅｌｄで、「Ｓｉｇｎ　Ｉｎ」と表示するためのＵＩ部品。
・雛形部品２６３２：Ｉｎｐｕｔの種別のＴｅｘｔＦｉｅｌｄで、アプリのユーザーネーム（ユーザーＩＤ）であるメールアドレスの入力を受け付けるためのＵＩ部品。すなわち、ユーザー特定情報の入力を受け付けるためのＵＩ部品。
・雛形部品２６３３：ボタンのＵＩ部品で、「ＣＲＥＡＴＥ　ＡＣＣＯＵＮＴ？」と表示する設定（ラベル）となっている。押下されたことに応じてテンプレート画面の一種である「Ｓｉｇｎ　Ｕｐ　ｆｏｒｍ」の画面（図２７（ｃ）に示す画面で、選択肢２６１４に対応するＵＩ画面）に遷移するというアクションが予め設定されている（予め定義されている）。
・雛形部品２６３４：ボタンのＵＩ部品で、「ＮＥＸＴ」と表示する設定（ラベル）となっている。押下されたことに応じて、雛形部品２６３２に入力された値（メールアドレス）を取得するとともに、テンプレ―ト画面の一種である「Ｓｉｇｎ　ｉｎ　Ｐａｓｓｗｏｒｄ」の画面（図２７（ａ）に示す画面で、選択肢２６１２に対応するＵＩ画面）に遷移するというアクションが予め定められている（予め定義されている）。
【０３４５】
図２７（ａ）に、「Ｓｉｇｎ　ｉｎ　Ｐａｓｓｗｏｒｄ」の画面のキャンバスにおける表示例を示す。なお、図２７（ａ）～図２７（ｋ）はそれぞれ、テンプレート画面をキャンバスに表示した場合のキャンバスの部分だけを示した部分的な表示例である。「Ｓｉｇｎ　ｉｎ　Ｐａｓｓｗｏｒｄ」の画面に配置される雛形部品２７１１～２７１５はそれぞれ以下のようなＵＩ部品である。
・雛形部品２７１１：Ｉｎｐｕｔの種別のＴｅｘｔＦｉｅｌｄで、アプリのユーザーパスワードの入力を受け付けるためのＵＩ部品。
・雛形部品２７１２：ボタンのＵＩ部品で、「Ｆｏｒｇｅｔ　Ｐａｓｓｗｏｒｄ？」と表示する設定となっている。押下されたことに応じて、テンプレ―ト画面の一種である「Ｒｅｓｅｔ　Ｐａｓｓｗｏｒｄ　Ｓｔｅｐ１」の画面（図２７（ｇ）に示す画面で、選択肢２６１８に対応するＵＩ画面）に遷移するというアクションが予め定められている（予め定義されている）。
・雛形部品２７１３：チェックボックスで、「Ｔｒｕｓｔ　ｔｈｉｓ　ｄｅｖｉｃｅ　ｆ

ｏｒ　１４　ｄａｙｓ」と表示される。
・雛形部品２７１４：「ＣＨＡＮＧＥ　ＥＭＡＩＬ」と表示されるボタンで、押下されたことに応じて「Ｓｉｇｎ　ｉｎ　ＩＤ」の画面に戻るというアクションが予め定められている（予め定義されている）。
・雛形部品２７１５：「ＳＩＧＮ　ＩＮ」と表示されるボタンで、押下されたことに応じて、雛形部品２６３２に入力された値（メールアドレス）と、雛形部品２７１１に入力された値（パスワード）と、雛形部品２７１３のチェックボックスのチェックの有無とを、当該アプリが構築されている実行環境に送信するというアクションが予め定められている（予め定義されている）。
【０３４６】
　実際に構築されたアプリにおいては、アプリユーザー用端末２０１に表示された雛形部品２７１５に対応するボタンの押下に応じて、アプリユーザー用端末２０１から、ユーザーネームとなるメールアドレス、パスワード、チェックの有無が実行環境に送信される。実行環境では、受信したメールアドレスとパスワードの組が正しいかを、マルチテナント実行環境のユーザー情報４１１（図１４（ｂ）に示したもの）またはシングルテナント実行環境のユーザー別情報（図１４（ｃ１）に示したもの）と一致するか照合する。照合の結果、認証ＯＫであれば、アプリへのログインがなされ、アプリユーザー用端末２０１にイニシャルＵＩが表示される。
【０３４７】
　同様に、以下の通りテンプレート画面のキャンバスにおける表示例を図示する。いずれも、アプリのユーザー認証に関する画面である。
・図２７（ｂ）：選択肢２６１３に対応するＵＩ画面。
・図２７（ｃ）：選択肢２６１４に対応するＵＩ画面。
・図２７（ｄ）：選択肢２６１５に対応するＵＩ画面。
・図２７（ｅ）：選択肢２６１６に対応するＵＩ画面。
・図２７（ｆ）：選択肢２６１７に対応するＵＩ画面。
・図２７（ｇ）：選択肢２６１８に対応するＵＩ画面。
・図２７（ｈ）：選択肢２６１９に対応するＵＩ画面。
・図２７（ｉ）：選択肢２６２０に対応するＵＩ画面。
・図２７（ｊ）：選択肢２６２１に対応するＵＩ画面。
・図２７（ｋ）：選択肢２６２２に対応するＵＩ画面。
【０３４８】
　上述した各テンプレート画面はカスタマイズが可能であり、図４のＳ４０５以降の処理で説明した通り、ＵＩエディタで開発者が編集可能である。編集可能な内容としては、雛形部品（雛形コンポーネント）とは異なる他のＵＩ部品の追加、雛形部品の配置位置、表示サイズの変更、雛形部品のプロパティの設定変更（表示される文字や色などの表示形態の変更）が可能である。また、テンプレート画面のキャンバスのアクションも設定・変更可能である。一方で、雛形部品のアクションの変更、削除、雛形部品の消去はできないものとする。また、テンプレート画面に追加した通常のＵＩ部品（雛形部品ではないＵＩ部品）に関しても、アクションの変更、削除は行えないものとする。このようにすることで、クラウドサービスを用いた認証処理に必要なアクションが修正されたことに起因する意図しないエラーが発生し、認証処理が上手く動作しなくなってしまうことを抑止することができる。すなわち、テンプレート画面では、ミスの無い認証に必要なアクションが雛形部品に予め定められており、修正不可となっているため、開発者がテンプレート画面のカスタマイズを行ったとしてもクラウドサービスを用いた認証処理は確実に上手く動作するようになっている。テンプレート画面に追加した通常のＵＩ部品にもアクションが設定できないようにしているのは、認証が上手く動作しなくなるようなアクションが設定されてしまうのを防止するためである。例えば、図２６（ｂ）の「Ｓｉｇｎ　ｉｎ　ＩＤ」のテンプレート画面に、新規にボタンを追加し、そのボタンに認証とは関係のないＵＩ画面に遷移するようなアクションが設定されてしまうと、認証に必要なメールアドレスとパスワ

ードを実行環境に送信することができなくなり、認証が動作しなくなってしまう。認証がＯＫとなっていない状態では遷移先のＵＩ画面も表示されないため、アプリは正常に動作しない。このような問題を、テンプレート画面に追加した通常のＵＩ部品にもアクションが設定できないようにすることで防止することができる。

【０３４９】

図２６（ｃ）に、テンプレート画面ではないＵＩ画面のキャンバスに配置されたボタン２６５４のコンテキストメニュー２６５５を表示した表示例を示す。このように、通常であれば、ボタンのコンテキストメニューにはアクション、消去の選択肢が含まれる。これは、図５（ｄ）の表示例で説明したことと同様である。従って、図７のＳ７１１～Ｓ７２０で説明した通り、プロパティの編集、アクションの編集、部品の消去といった編集が可能である。

【０３５０】

図２６（ｄ）に、テンプレート画面のキャンバスに配置されたボタン２６３４のコンテキストメニュー２６３５を表示した表示例を示す。このように、テンプレート画面に配置された雛形部品のコンテキストメニューにはアクション、消去の選択肢が表示されない。これによって、アクションの設定、雛形部品の消去ができないように制御している。すなわちアクションボードも表示されない。従って、図７のＳ７２１～Ｓ７２４で説明した通り、雛形部品についてはプロパティの編集（表示形態の変更）のみが行える。

【０３５１】

図２６（ｅ）に、図２６（ｂ）に示したテンプレート画面のキャンバスに対して編集（カスタマイズ）を加えた後の表示例を示す。図２６（ｂ）と図２６（ｅ）とで同じＵＩ部品には同じ符号を付す。ＵＩ部品２６４１は、開発者によって追加された出力の種別のＵＩ部品であり、図１２のＳ１２１７のファイル管理操作でアップロードされた画像ファイルを指定して表示したものである。この画像を、アプリを表すアイコンや画像にすれば、この認証画面がどのアプリの認証画面なのかをユーザーが把握しやすくなる。ＵＩ部品２６４２は、開発者によって追加された、テキストメッセージを出力する種別のＵＩ部品である。また、この画面のキャンバスのアクションとして、今日の日付を取得して、月末であれば「月末締め日ですので、申請をお忘れなく」という文字列をＵＩ部品２６４２に表示する、というアクションを、キャンバスのアクションボードを開いて開発者が設定したものとする。このように、この認証用の画面を表示する際の条件によって表示内容が変更されるＵＩ部品（コンポーネント）を配置することも可能である。雛形部品２６３１は、図２５（ｂ）の雛形部品２６３１と同じＵＩ部品（同じＩＤのＵＩ部品）であるが、開発者のプロパティの編集操作によって表示する内容（ラベル）が変更され、「Ｌｏｇｉｎ」と表示される。このように、認証に必須のアクション以外については画面設計の自由度が高い。従って例えば、アプリのユーザーが、何のアプリの認証画面（ログイン画面）なのかを容易に認識可能な画面にしたり、社内で取り決めた社内標準デザインに沿ったデザインにしたり、アプリの性質に合わせたより好適なデザインの画面にしたりすることが可能となる。

【０３５２】

＜プロパティボックスの表示例＞

キャンバスに配置されたＵＩ部品のコンテキストメニューからプロパティを選んで表示されるプロパティボックスについて説明する。

【０３５３】

図２８（ａ）に、パイチャートのプロパティボックスの表示例を示す。図２８は、図５（ｂ）に示した表示状態から、キャンバス５３０に配置されたＵＩ部品であるパイチャート５３１を右クリックして表示されたコンテキストメニューのうち、プロパティを選択することで表示する画面である。この画面を表示する処理と、表示されたプロパティボックスに対する編集処理の受付は、前述した図７のＳ７０６で行われる。プロパティボックス２８１０が、パイチャート５３１のプロパティボックスである。プロパティボックス２８１０は、キャンバス５３０に表示されたパイチャート５３１とともに表示される。パイチ

ャート５３１に表示される中身である円グラフの比率や色分けなどの表示形態は、プロパティボックス２８１０に設定された内容に基づいて表示される。プロパティボックス２８０１で設定内容が変更され、適用ボタン２８１１が押下されると、プロパティボックス２８０１の最新の設定値（変更された設定値）に基づいて、パイチャート５３１の表示形態が更新される。すなわち、開発者は、パイチャート５３１の表示形態がどのように変化するのかを確認しながら、プロパティボックス２８１０に対する設定操作を行うことができる。

【０３５４】

ＵＩ部品一覧からパイチャートをドラッグ＆ドロップでキャンバスに配置した際、何の値も設定されていないと、円グラフの比率毎の区域が表示されないため、ただの円のような表示となってしまう。これでは、開発者が配置されたＵＩ部品を見て、それがパイチャート（円グラフ）であることを直感的に認識することができない。これを解決するため、パイチャートのプロパティとして予め定められた初期値（開発者が設定したものではないデフォルトの設定値）を設定しておくものとする。こうすることで、ＵＩ部品一覧からパイチャートをドラッグ＆ドロップでキャンバスに配置した際、最初から比率別に区域分けされた円グラフが表示されるため、開発者は、配置したＵＩ部品が円グラフであることを素早く直感的に把握することができる。

【０３５５】

プロパティボックス２８１０のうち、Ｖａｌｕｅ入力領域２８１２は、パイチャート５３１が示す各区域のＩＤ、数値、色、ラベル（表示文字列）を入力する領域である。開発者が何も設定していない状況であれば、初期値が図示のように表示される。図示の例では、初期値の一部だけが表示されており、スクロールすることで最初から最後までを見ることができる。Ｖａｌｕｅ入力領域２８１２に表示される初期値の全文は以下の通りであるものとする。なお、「”」はダブルクォーテーションであるものとする

【０３５６】

[｛”ｉｄ”：”Ｊａｎ”，”ｖａｌｕｅ”：開発者用端末１００．３３，”ｃｏｌｏｒ”：”＃３９４Ｅ７９”，”ｍｏｎｔｈ”：”Ｊａｎ”｝，｛”ｉｄ”：”Ｆｅｂ”，”ｖａｌｕｅ”：２２．１２，”ｃｏｌｏｒ”：”＃５Ｅ８３ＢＡ”，”ｍｏｎｔｈ”：”Ｆｅｂ”｝，｛”ｉｄ”：”Ｍａｒ”，”ｖａｌｕｅ”：５３．２１，”ｃｏｌｏｒ”：”＃Ｃ２Ｄ２Ｅ９”，”ｍｏｎｔｈ”：”Ｍａｒ”｝，｛”ｉｄ”：”Ａｐｒ”，”ｖａｌｕｅ”：３４．２５，”ｃｏｌｏｒ”：”＃９Ａ８ＢＡ５”，”ｍｏｎｔｈ”：”Ａｐｒ”｝，｛”ｉｄ”：”Ｍａｙ”，”ｖａｌｕｅ”：２４．６５，”ｃｏｌｏｒ”：”＃Ｅ３Ｃ５Ｄ５”，”ｍｏｎｔｈ”：”Ｍａｙ”｝]

Ｖａｌｕｅ入力領域２８１０に表示された初期値は、プログラミング言語で記述する際の構造で表示されている。上記の初期値の全文のうち、「｛」と「｝」で囲まれた部分がパイチャートにおいて１つの区域に対応する１つのデータセットである。「”」で囲まれた文字列が「：」で対応付けられた１組の文字列は、キー名とキーの値の組である。例えば、「”ｉｄ”：”Ｊａｎ”」では、キー名が「ｉｄ」であり、キーの値が「Ｊａｎ」である。「，」が前のキーと次のキーとの区切りである。すなわち、上記の初期値の全文のうち、｛”ｉｄ”：”Ｊａｎ”，”ｖａｌｕｅ”：開発者用端末１００．３３，”ｃｏｌｏｒ”：”＃３９４Ｅ７９”，”ｍｏｎｔｈ”：”Ｊａｎ”｝という複数組分を含む部分が１つのデータセット（１セット）であり、ｉｄ、ｖａｌｕｅ、ｃｏｌｏｒ、ｍｏｎｔｈの４つのキー名と値の組が含まれている。同様のデータセットが全部で５つあることがわかる。このように、パイチャートの初期値は、それぞれデフォルトの設定値群を含むデータセットを複数含む設定値のセットである。

【０３５７】

開発者は、プロパティボックス２８１０のＶａｌｕｅ入力領域２８１２に表示された初期値を、Ｖａｌｕｅ入力領域２８１２を選択してキーボードからの入力を行うことで自由に編集可能である。ただし、初期値として、上記のように構造がわかるように表示されているため、どこをどのように編集すれば良いのかを理解しやすい。また、適用ボタン２８

１１を押下すれば編集した内容がパイチャート５３１に反映されるため、どこの構造の値を変更するとパイチャート５３１がどのように変化するかを確認できる。そのため、プログラミング言語で記述可能な構造で表示された文字列のうち、どこがどういう意味なのかというのも理解しやすい。

【０３５８】

単にパイチャートの設定値を設定できるようにするのであれば、プロパティボックス２８１０において「区域１のＩＤ」、「区域１の値」、「区域１の色」、「区域１の表示ラベル」…といったように設定項目を分け、それぞれに対して開発者に入力あるいは選択させるという設定方法も考えられる。しかし、本実施形態ではそうはせず、プログラミング言語で記述する際の構造で初期値を表示し、開発者が編集する場合もプログラミング言語で記述する際の構造に沿った記述で入力させる。これは、パイチャートの設定値を、他のＵＩ部品またはキャンバスのアクションボードにＪａｖａＳｃｒｉｐｔで入力するアクションによって変更可能であるためである。初期値をプログラミング言語で記述可能な構造で、コピー可能なテキスト（文字列）で表示しておけば、開発者は、この文字列を公知のコピーによってクリップボードにコピーし、アクションボードにペーストすることができる（コピーアンドペーストすることができる）。そして、アクションボード上で変更を加えたい部分の文字だけを修正すれば、パイチャートの表示内容を変更するアクションを容易に記述することができる。このようにすることで、開発者が、パイチャートの表示形態を変更するようなアクションを記述したい場合に、どのように記述すればよいかわからないために意図通りのアクションを記述できない、あるいは記述の仕方を調べるのに多大な時間を費やしてしまうことを抑制できる。すなわち、より効率的なソフトウェア開発を行うことができる。プログラミング言語であるため、その他の編集を加えたい場合にも自由度高く編集することができる。

【０３５９】

図２９に、開発者が配置した、パイチャート５３１とは異なる他のＵＩ部品（ボタン）のアクションボードに、開発者によって、パイチャート５３１の表示形態（表示内容）を変更（指定）するアクションが入力された場合のアクションボードの表示例を示す。図示のアクションボード２９００は、構築後のアプリにおいてボタンが押下された場合に実行すべきアクションをプログラミング言語で記述する記述領域（入力領域）である。アクションボード２９００に表示された文字列（ＪａｖａＳｃｒｉｐｔでの文字列）のうち、２行目から３２行目は、プロパティボックス２８１０のＶａｌｕｅ入力領域２８１２に表示された初期値をコピーアンドペーストし、見やすさのための改行を加えたものをベースとして、数値２９０１～数値２９５１だけを編集（変更）したものである。もちろん、他の部分を変更してもよいし、データセット（「｛｝」で囲まれた部分）の追加や削除を行っても良い。このように、プログラミング言語でのアクションの入力が非常に容易となる。構築後のアプリにおいて、このアクションボードに対応するボタンが押下された場合、パイチャートは、初期値ではなく、このアクションボードに記述された内容に基づいた表示形態で表示される。

【０３６０】

パイチャートと同様に、ＵＩ部品のうちリストにも初期値（デフォルトの設定値）が設定されており、プロパティボックスにおいて初期値がプログラミング言語で記述する場合の構造に則って表示される。図２８（ｂ）に、リスト２８０２のプロパティボックス２８２０の表示例を示す。Ｖａｌｕｅ入力領域２８２１には、図示の通り、初期値がプログラミング言語で記述する場合の構造に則って表示される。

【０３６１】

パイチャートと同様に、ＵＩ部品のうちＬＩＮＥ　ＣＨＡＲＴ（ラインチャート、線グラフ）にも初期値（デフォルトの設定値）が設定されており、プロパティボックスにおいて初期値がプログラミング言語で記述する場合の構造に則って表示される。図２８（ｃ）に、ＬＩＮＥ　ＣＨＡＲＴ２８０３のプロパティボックス２８３０の表示例を示す。入力領域２８３１～２８３４には、図示の通り、初期値がプログラミング言語で記述する場合

の構造に則って表示される。
【０３６２】
パイチャートと同様に、ＵＩ部品のうちデータグリッド（表）、コンボボックス、タブ部品、ステッパー、パンくずリスト（Ｂｒｅａｄｃｒｕｍｂｓ）にも初期値（デフォルトの設定値）が設定されており、プロパティボックスにおいて初期値がプログラミング言語で記述する場合の構造に則って表示される。図２８（ｄ）に、データグリッド２８０４のプロパティボックス２８４０の表示例を示す。入力領域２８４１には、図示の通り、初期値がプログラミング言語で記述する場合の構造に則って表示される。データグリッドの初期値は、データグリッドの１行毎の各カラムの値を１セットのデータセットとして、複数行分のデータセットを含む。
【０３６３】
＜データグリッドのプロパティとアクションの設定＞
図３０に、開発者用端末１００におけるデータグリッドのコンテキストメニュー処理のフローチャートを示す。この処理は、前述した図７のＳ７０５の詳細フローチャートである。この処理は開発者用端末１００が実行する。
【０３６４】
Ｓ３００１では、開発者用端末１００は、データグリッド用のコンテキストメニューを表示する。図３１（ａ）に、データグリッド用のコンテキストメニューの表示例を示す。キャンバス５３０に配置されたＵＩ部品であるデータグリッド３１１０は、カラム３１１１～３１１６の６つのカラムを有している。なお、データグリッド３１１０を囲む点線はデータグリッド３１１０の全体を示すために図示した図面上の補助線であって、表示されるものではない。マウスカーソルの近傍に、データグリッド用のコンテキストメニュー３１２０が表示される。コンテキストメニュー３１２０には、メニュー項目となる選択肢として、少なくともプロパティ３１２１とアクション３１２２が表示される。
【０３６５】
Ｓ３００２では、開発者用端末１００は、コンテキストメニュー３１２０に含まれる選択肢のうち、プロパティ３１２１が押下（選択）されたか否かを判定する。プロパティ３１２１が押下（選択）された場合はＳ３０１０に進み、そうでない場合にはＳ３００３へ進む。
【０３６６】
Ｓ３００３では、開発者用端末１００は、コンテキストメニュー３１２０に含まれる選択肢のうち、アクション３１２２が押下（選択）されたか否かを判定する。アクション３１２２が押下（選択）された場合はＳ３０２０に進み、そうでない場合にはＳ３００４へ進む。
【０３６７】
Ｓ３００４では、開発者用端末１００は、コンテキストメニュー３１２０に含まれる選択肢のうち、その他の選択肢が押下（選択）されたか否かを判定する。その他の選択肢が押下（選択）された場合はＳ３００５に進み、そうでない場合にはＳ３００６へ進む。
【０３６８】
Ｓ３００５では、開発者用端末１００は、その他の選択肢の押下（選択）に応じたその他の処理を行う。例えば、消去する選択肢が選択された場合には指定ＵＩ部品であるデータグリッドをキャンバスから削除（消去）する。
【０３６９】
Ｓ３００６では、開発者用端末１００は、コンテキストメニューを閉じる操作（例えば、コンテキストメニューが表示された領域外をクリックする操作）があったか否かを判定する。コンテキストメニューを閉じる操作があった場合には、Ｓ３００７でコンテキストメニューを非表示とし、図３０の処理を終了する。閉じる操作がない場合にはＳ３００２に戻る。
【０３７０】
Ｓ３０１０では、開発者用端末１００は、指定ＵＩ部品であるデータグリッドのプロパ

ティボックスを表示する。図３１（ｂ）～図３１（ｆ）に、データグリッドのプロパティボックスの表示例を示す。図３１（ｂ）～図３１（ｆ）において、同じ表示アイテムには同じ符号を付す。図３１（ｂ）の適用ボタン３１３１は、プロパティボックスに入力された内容（変更された設定内容）を反映することを指示するボタンアイコンである。適用ボタン３１３１が押下されると、プロパティボックスの最新の設定内容に基づいて、プロパティボックスとともに表示しているキャンバスに配置された指定ＵＩ部品であるデータグリッドの表示形態を変更する。カラム選択欄３１３２は、設定を変更したいカラムを選択するための選択タブである。データグリッドのプロパティボックスでは、カラム毎に設定を受け付ける画面を表示し、カラム毎のプロパティの設定を受け付ける。カラム追加ボタン３１３３は、指定ＵＩ部品であるデータグリッドにカラムを追加することを指示する表示アイテムである。

【０３７１】

Ｓ３０１１では、開発者用端末１００は、プロパティボックス内のカラム選択欄３１３２でいずれかのカラムに対応するタブが押下されたか否かを判定する。すなわち、設定対象としていずれかのカラムが選択されたか否かを判定する。いずれかのカラムに対応するタブが押下された場合にはＳ３０１２に進み、そうでない場合にはＳ３０１３に進む。なお、カラム選択欄３１３２に表示されたタブ（選択肢）のうち、最も左側に配置されるタブ（不図示）は、「ＤａｔａＧｒｉｄ」と表記されたタブであり、指定ＵＩ部品であるデータグリッドの全体に関する設定を行う設定画面を表示させるためのタブである。この「ＤａｔａＧｒｉｄ」のタブだけは、特定のカラムに関するものではない（特定のカラムを選択するものではない）。

【０３７２】

Ｓ３０１２では、開発者用端末１００は、プロパティボックスのうちカラム選択欄３１３２より下部の領域の表示内容を、選択されたカラムに関する設定画面（要素画面）に切り替える。図３１（ｃ）に、カラム選択欄３１３２のうち、タブ３１３２Ｅ（「ＣｏｍｐａｎｙＥ」と表記されたタブ）が選択された場合の表示例を示す。プロパティボックスのうちカラム選択欄３１３２より下側の領域には、タブ３１３２Ｅに対応するカラムの設定を受け付ける画面として、タブ３１３２Ｅの設定画面の上部領域３１３４ｔが表示される。ここには例えば、カラムの部品種別（コンポーネントのタイプ）を選択する種別設定欄３１３５が含まれる。タブ３１３２Ｅの設定画面は下側にさらに続いており、スクロールすることでさらに下側の方を表示することができる。図３１（ｄ）に、スクロールさせて、タブ３１３２Ｅの設定画面の下部領域３１３４ｄを表示させた場合の表示例を示す。下部領域３１３４ｄには、タブ３１３２Ｅに対応するカラム自体の削除を指示するためのカラム削除ボタン３１３６と、タブ３１３２Ｅの設定画面に設定した内容を適用してキャンバスに表示されているデータグリッドに反映する指示を行う適用ボタン３１３７とが含まれる。カラム選択欄３１３２はスクロールに伴って上側に移動し、プロパティボックス内で見えなくなっているが、上部領域３１３４ｔが表示されるようにスクロールを戻せば再度表示される。なお、Ｓ３０１１で「ＤａｔａＧｒｉｄ」のタブが選択された場合は、プロパティボックスのうちカラム選択欄３１３２より下部の領域の表示内容を、データグリッドの全体に関する設定を行う設定画面に切り替える。

【０３７３】

Ｓ３０１３では、開発者用端末１００は、プロパティボックス内のカラム追加ボタン３１３３が押下されたか否かを判定する。カラム追加ボタン３１３３が押下された場合にはＳ３０１４に進み、そうでない場合にはＳ３０１５に進む。

【０３７４】

Ｓ３０１４では、開発者用端末１００は、カラムの追加処理を行う。カラムの追加処理では、まず、図３１（ｅ）に示すカラムの追加フォーム３１３８を表示し、開発者（ユーザー）から、追加するカラム（表の列）のＩＤとラベル（表示するカラムの名前）の設定を受け付ける。そして、ＡＤＤ　ＴＯ　Ｖａｌｕｅボタンが押下されると、新しいカラムが追加され、カラム選択欄３１３２に新しいカラムに対応するタブが追加される。

【０３７５】
図３１（ｆ）に、カラムが追加された場合のプロパティボックスの表示例を示す。図３１（ｃ）と比較して、カラム選択欄３１３２には、新しいカラムに対応するタブ３０３２Ｆ（「ＣｏｍｐａｎｙＦ」と表記されたタブ）が追加されている。追加した直後は追加したカラムが選択され、カラム選択欄３１３２より下側の領域には、追加したカラムに対応するタブ３０３２Ｆの設定画面の上部領域３１３９ｔが表示されている。このように、本実施形態では、データグリッド（表）へのカラムの追加は、データグリッドの設定画面であるプロパティボックスに対する操作によって行われる。
【０３７６】
Ｓ３０１５では、開発者用端末１００は、選択されているカラムについての設定操作があったか否かを判定する。具体的には、カラム選択欄３１３２より下側の領域に表示された設定画面への各種設定欄への入力操作があったか否かを判定する。設定欄への入力操作があった場合にはＳ３０１６へ進み、そうでない場合はＳ３０１７へ進む。
【０３７７】
Ｓ３０１６では、開発者用端末１００は、設定操作を受け付け、表示に反映する。例えば、種別設定欄３１３５に対する種別の設定を受け付ける。このように、データグリッドは、カラム毎にコンポーネント（部品）としての種別を選択できる。カラムに設定できるコンポーネントの種別の選択肢として、ＴｅｘｔＩｎｐｕｔ、ＮｕｍｂｅｒＩｎｐｕｔ、ＣｏｍｂｏＢｏｘ、Ｍｕｌｔｉ－Ｓｅｌｅｃｔ、ＣｈｅｃｋＢｏｘ、ＤａｔａＰｉｃｋｅｒ、Ｌｉｎｋ、Ｂｕｔｔｏｎ、ＩｃｏｎＢｕｔｔｏｎが表示され、開発者はいずれかを選択して設定できる。また、その他の設定可能な項目（設定項目）の少なくとも一部が、選択されたコンポーネントの種別に応じて変わる。例えば、種別としてＴｅｘｔＩｎｐｕｔを選択した場合は、他の設定項目には以下が含まれる。ＶａｌｕｅＴｙｐｅ、ＩＤ，Ｌａｂｅｌ（カラム名として表示される文字列）、Ｗｉｄｔｈ（カラムの幅）、ＮｕｍｂｅｒＦｏｒｍａｔ、ＤａｔａＦｏｒｍａｔ、Ｏｐｔｉｏｎｓ、Ｆｏｏｔｅｒ、ＦｏｏｔｅｒＴｅｘｔ、Ｓｏｒｔｉｎｇ、Ｖｉｓｉｂｉｌｉｔｙ。また、「ＤａｔａＧｒｉｄ」のタブに対応するデータグリッドの全体に関する設定画面で設定可能な設定項目には、以下が含まれる。ＩＤ（データグリッドのＵＩ部品ＩＤ）、ＤａｔａＧｒｉｄ全体のサイズ（幅、高さ）、デプロイ後のアプリにおいて編集可能とするか、デプロイ後のアプリにおいて当該データグリッドに行追加を指示する行追加ボタンを表示するか、データグリッドの削除ボタンを表示するか、データグリッドの更新ボタンを表示するか、等。
【０３７８】
Ｓ３０１７では、開発者用端末１００は、適用ボタン３１３１または適用ボタン３１３７が押下されたか否かを判定する。適用ボタン３１３１が押下された場合にはＳ３０１８に進み、そうでない場合にはＳ３０１９へ進む。
【０３７９】
Ｓ３０１８では、開発者用端末１００は、プロパティボックスにおいて設定された内容をメモリ１０２に保持している定義情報に記録するとともに、設定された内容を反映してＵＩエディタのキャンバスに表示された指定ＵＩ部品であるデータグリッドの表示形態を変更する。例えば、カラムが追加されていれば１列追加した表示形態でデータグリッドを表示し、カラムの幅が変更されていればキャンバス上のデータグリッドのカラムの幅を変更して表示する。
【０３８０】
Ｓ３０１９では、開発者用端末１００は、終了ボタン３１４０が押下されたか否かを判定する。終了ボタン３１４０が押下された場合は、プロパティボックスを非表示とし、図３０の処理を終了する。終了ボタン３１４０が押下されていない場合はＳ３０１１に進む。
【０３８１】
このように、データグリッド（表）に対するカラムの追加と、カラム毎の設定操作とを、より操作性良く行うことができる。特に、データグリッドに対してカラムの追加をする

操作を、データグリッドのプロパティボックスに対する操作で行え、そのまま同じプロパティボックス内で追加したカラムに関する設定操作を行える。すなわち、カラムの追加と、追加したカラムの設定操作という一連の操作を、プロパティボックスへの操作という同様の操作感でスムーズに行うことができる。また、カラムを追加した後に、カラムのコンポーネントとして設定可能な選択肢の中から種別の設定を行うため、混乱なくカラムの追加と種別の設定とを行うことができる。

【０３８２】

Ｓ３０２０では、開発者用端末１００は、コンテキストメニューのサブメニューとして、カラム別のアクションの選択肢を表示する。図３２（ａ）に、Ｓ３０２０での表示例を示す。図３２（ａ）は、図３１（ａ）の状態からアクション３１２２が押下された場合の表示例である。図３１（ａ）と図３２（ａ）とで同じ表示物には同じ符号を付す。サブメニュー３２１０がコンテキストメニュー３１２０のアクション３１２２のサブメニューとして表示される。サブメニュー３２１０には、選択肢３２１１～３２１７が含まれる。選択肢３２１１は、データグリッド３１１０の全体に対する操作に応じて実行するアクションのアクションボードを開くための選択肢である。選択肢３２１２～３２１７は、データグリッド３１１０の各カラムが操作された場合に実行するカラム毎のアクションのアクションボードを開くための選択肢である。選択肢３２１２～３２１７はそれぞれ、カラム３１１１～３１１６に対応する。このとき、指定ＵＩ部品のデータグリッドと異なるデータグリッドのカラムのアクションボードに対応する選択肢は表示しない。

【０３８３】

Ｓ３０２１では、開発者用端末１００は、データグリッド全体のアクションの選択肢（選択肢３２１１）またはカラム別のアクションの選択肢（３２１２～３２１７）のうちいずれかが選択されたか否を判定する。いずれかのアクションの選択肢が選択された場合にはＳ３０２２に進み、そうでない場合にはＳ３０２１で選択を待つ。

【０３８４】

Ｓ３０２２では、開発者用端末１００は、選択されたアクションの選択肢に対応するアクションボードを表示する。

【０３８５】

図３２（ｂ）に、データグリッド全体のアクションの選択肢（選択肢３２１１）に対応するアクションボードの表示例を示す。全体またはカラムの選択領域３２２０で「ＤＡＴＡ　ＧＲＩＤ」が選択されており、アクションボード３２１１ａがデータグリッド全体のアクションを入力する領域であることを示している。アクションボード３２１１ａで設定された内容を実行するトリガーは予め定められており、構築済みアプリにおいて指定ＵＩ部品であるデータグリッドの領域のカラム外に表示される全体に関するボタンの押下がトリガーとなる。言い換えれば、このトリガーは、指定ＵＩ部品であるデータグリッド（表）に関するトリガーであって、当該データグリッドのカラムにかかわらないトリガーである。全体に関するボタンとは、前述したプロパティボックスの「ＤａｔａＧｒｉｄ」のタブに対応する設定画面への設定で、表示すると設定したことにより表示されるボタンである。例えば前述したデータグリッドの削除ボタン、データグリッドの更新ボタンなどである。従って開発者は、アクションを実行するトリガーが何かというのは設定する必要がない（ＪａｖａＳｃｒｉｐｔで記述する必要はない）。このアクションボード３２１１ａに例えば、特定のカラムの全ての行の合計を算出して他のＵＩ部品内に表示するというアクション、データグリッドに表示された内容をデータベースに保存するというアクション、他のＵＩ画面に画面遷移するというアクション等をＪａｖａＳｃｒｉｐｔで記述することで設定可能である。

【０３８６】

図３２（ｃ）に、データグリッドの特定のカラムのアクションの選択肢（選択肢３２１２）に対応するアクションボードの表示例を示す。選択領域３２２０で「ＭＯＮＴＨ」が選択されており、アクションボード３２１２ａがラベルまたはカラムＩＤが「ＭＯＮＴＨ」のカラム（カラム３１１１）のアクションを入力する領域であることを示している。カ

ラムのアクションボード３２１２ａで設定された内容を実行するトリガーは予め定められており、構築済みアプリにおいて、指定ＵＩ部品のうち、対応するカラムのいずれかの行のセルの値が変更された、あるいは、対応するカラムのいずれかの行のセルに表示されたボタンが押下されたことがトリガーとなる。従って開発者は、アクションを実行するトリガーが何かというのは設定する必要がない（ＪａｖａＳｃｒｉｐｔで記述する必要はない）。このアクションボード３２１２ａに例えば、アクションボード３２１２ａに対応するカラムの変更のあった行の値と、同じ行の他の第１のカラムの値との合計を算出し、同じ行の他の第２のカラムに表示するといったアクションを設定可能である。このアクションの例は、表の１列目の数値に変更があった場合に、同じ行の３列目に、変更後の１列目の数値と２列目の数値の合計を表示する、といったアクションである。すなわち、選択した選択肢３２１２に対応するカラムとは異なる、指定ＵＩ部品であるデータグリッドのうちの他のカラムに影響を与えるアクションの設定を受け付け可能である。

【０３８７】

なお、選択領域３２２０で他のカラムを選択することで、アクションボードを開いた後にも、他のカラムのアクションボードに切り替えて表示することが可能である。

【０３８８】

Ｓ３０２３では、開発者用端末１００は、表示されたアクションボードに対するアクションの入力操作を受け付ける。この処理は、前述した図８のＳ８０２～Ｓ８２３の処理と同様である。

【０３８９】

Ｓ３０２４では、開発者用端末１００は、アクションボードを閉じる操作（アクションボード処理を終了する操作）があったか否かを判定する。アクションボードを閉じる操作が無い場合にはＳ３０２３に進んで処理を繰り返す。アクションボードを閉じる操作があった場合にはアクションボードを非表示とし、選択中ＵＩ画面のキャンバスの表示に切り替え、ＵＩエディタ処理へ戻る。

【０３９０】

このように、データグリッド（表）に含まれる各カラムに対して、より操作性良くアクションの設定を行うことができる。特に、サブメニュー３２１０に示したように、データグリッド（表）に含まれる全てのカラムのアクションボードに対応する選択肢を一覧表示するため、開発者は、カラム毎にアクションを設定可能であることを認識することができ、カラムに対するアクションの設定漏れなどが発生する可能性を低減することができる。また、指定ＵＩ部品であるデータグリッドのコンテキストメニューのサブメニューとして表示されるため、指定ＵＩ部品であるデータグリッドとの関係を明確に把握できる。すなわち、アクションボードで設定するアクションがどのデータグリッドのどのカラムに関するものであるかというのを混乱なく把握しながら設計作業を行うことができる。

【０３９１】

＜アクション実行関連処理＞

図３３（ａ）に、定義情報の記録先となる開発環境３００が実行する保存処理（記録制御処理）のフローチャートを示す。この処理は、前述の図４のＳ４２２において、開発者用端末１００から送信された、定義情報を受信したことに連動して実行する処理である。

【０３９２】

また、図３４に、開発者用端末１００、開発環境３００、実行環境４００、アプリユーザー用端末２００、２０１に記録される情報の遷移を図示する。図３４は、図３３（ａ）～図３３（ｃ）のフローチャートの処理による情報の遷移の様子を模式的に表した図である。

【０３９３】

図３３（ａ）のＳ３３０１では、開発環境３００は、開発者用端末１００から選択中アプリの定義情報（ＵＩ定義情報）を受信したか否かを判定する。ここで受信する定義情報は、前述の図４のＳ４２２において、開発者用端末１００から送信されたものである。ＵＩ定義を受信した場合にはＳ３３０２に進み、そうでない場合にはＳ３３０１で待機する

。

【０３９４】

図３４において、開発者用端末１００のメモリ１０２に記録されているＵＩ定義情報３４０１が、Ｓ３３０１で受信する定義情報である。本実施形態では、ＵＩ定義情報は「ｕｉＤｅｆ．ｊｓｏｎ」というファイル名の、ＪＳＯＮフォーマット で記述された、 テキスト形式のファイルであるものする。ＪｓｏｎはＪａｖａＳｃｒｉｐｔ Ｏｂｊｅｃｔ Ｎｏｔｉｆｉｃａｔｉｏｎの略であり、ＪａｖａＳｃｒｉｐｔで値を取り扱うためのドキュメント規格であり、データ記述言語である。ＵＩ定義情報３４０１には、イニシャルＵＩ等のアプリに関する各種設定内容や、ＵＩ画面毎のＵＩコンポーネント（ＵＩ部品）の情報（配置位置、サイズ、色など）を定義する情報が含まれる。また、ＵＩ定義情報３４０１には、アクション記述部分３４０２が含まれている。このアクション記述部分３４０２は、各ＵＩ部品やキャンバスのアクションボードや関数の設定画面（創作関数の設定画面を含む）に入力された文字列である。アクション記述部分３４０２には、ＪａｖａＳｃｒｉｐｔで記述されたアクションと、ＪａｖａＳｃｒｉｐｔを入力することなく関数の設定画面で設定された内容を記述したＪｓｏｎフォーマットの関数定義などが含まれる。このＵＩ定義情報３４０１が開発者用端末１００から開発環境３００に送信され（アップロードされ）、Ｓ３３０１で開発環境３００によって受信される。

【０３９５】

Ｓ３３０２では、開発環境３００は、ＵＩ定義情報をストレージ３２０のうち、ログイン開発者用の領域に保存する。この結果、図３４に示す通り、開発環境３００にＵＩ定義情報３４１１が記録される。この保存の直後においては、開発環境３００に記録されたＵＩ定義情報３４１１は、開発者用端末１００に記録されたＵＩ定義情報３４０１と同じ情報である。

【０３９６】

Ｓ３３０３では、開発環境３００は、Ｓ３３０２で保存したＵＩ定義情報からアクション情報を抽出する。すなわち、開発環境３００は、図３４のＵＩ定義情報３４１１からアクション記述部分３４１２を抽出する。

【０３９７】

Ｓ３３０４では、開発環境３００は、Ｓ３３０３で抽出したアクション記述部分３４１２から、実行環境用プログラムを生成して、ストレージ３２０のうちログイン開発者用の領域に保存する。すなわち、図３４のＵＩ定義情報３４１１から抽出したアクション記述部分３４１２に基づいて、実行環境用プログラム３４１３を生成して開発環境内に保存する。実行環境用プログラム３４１３はＪａｖａＳｃｒｉｐｔで記述されたテキストデータである。実行環境用プログラム３４１３は、アクション記述部分３４１２から取得したアクションボードに入力されていた文字列に加えて、実行環境の実行エンジンで実行するために必要な補充部分を追加したプログラムである。

【０３９８】

図３３（ａ）で説明した保存処理によって開発環境３００に保存されたＵＩ定義情報３４１１と実行環境用プログラム３４１３が、デプロイ処理によって実行環境４００にデプロイ（配置、保存、記録、構築）される。この処理は、前述した図２４のＳ２４２３またはＳ２４２４で行われる。これによって、図３４に図示した通り、実行環境４００にＵＩ定義情報３４２１と実行環境用プログラム３４２３が記録される。実行環境４００にＵＩ定義情報３４２１と実行環境用プログラム３４２３はそれぞれ、ＵＩ定義情報３４１１と実行環境用プログラム３４１３と同じ情報である。この状態が、アプリ生成された状態である。

【０３９９】

図３３（ｂ）に、実行環境４００におけるアプリ実行処理のフローチャートを示す。この処理は、実行環境４００の実行エンジンが実行する処理であり、アプリユーザー用端末２００または２０１から、デプロイ済み（構築済み、生成済み）のアプリケーションに対するアクセスがあった場合に実行する処理である。

【０４００】
図３３（ｃ）に、アプリユーザー用端末２００または２０１おけるアプリ実行処理のフローチャートを示す。この処理は、アプリユーザー用端末２００または２０１のＣＰＵ１０１が実行する処理であり、アプリユーザー用端末２００または２０１のブラウザソフトで、実行環境４００にデプロイ済み（構築済み、生成済み）のアプリケーションに対してアクセスした場合に実行する処理である。また、図３３（ｂ）の実行環境４００におけるアプリ実行処理と、図３３（ｃ）のアプリユーザー用端末２００または２０１におけるアプリ実行処理とは、連動して行われる処理である。以下、アプリユーザー用端末２００または２０１による処理について、代表してアプリユーザー用端末２００が実行するものとして説明する（アプリユーザー用端末２０１に関する説明は同様であるため省略する）。

【０４０１】
図３３（ｂ）のＳ３３１１では、実行環境４００は、アプリユーザー用端末２００から送信されたＵＩ定義情報の取得要求を受信したか否かを判定する。ここで受信するＵＩ定義情報の取得要求は、後述する図３３（ｃ）のＳ３３３３でアプリユーザー用端末２００から送信されるものである。ＵＩ定義情報の取得要求を受信した場合はＳ３３１２に進み、そうでない場合はＳ３３１１で待機する。

【０４０２】
Ｓ３３１２では、実行環境４００は、ＵＩ定義情報をアプリユーザー用端末２００に送信する。これによって、図３４に図示した通り、実行環境４００に記録されたＵＩ定義情報３４２１がアプリユーザー用端末２００にダウンロードされ、ＵＩ定義情報３４３１として記録される。ＵＩ定義情報３４２１とＵＩ定義情報３４３１とは同じ情報である。

【０４０３】
Ｓ３３１３では、実行環境４００は、アプリユーザー用端末２００からアクション要求を受信したか否かを判定する。アクション要求とは、アクションボードに入力されたアクションの内容を実行する要求である。ここで受信するアクション要求は、後述する図３３（ｃ）のＳ３３４５でアプリユーザー用端末２００から送信されるものである。アクション要求を受信した場合はＳ３３１４に進み、そうでない場合にはＳ３３２０に進む。

【０４０４】
Ｓ３３１４では、実行環境４００は、アプリユーザー用端末２００から入力項目の値を受信したか否かを判定する。入力項目の値とは、アプリの画面に表示されるＵＩ部品のうち、入力項目に分類されるＵＩ部品に対して、アプリユーザー用端末２００においてユーザーによって入力された値である。例えば、Ｔｅｘｔｆｉｅｌｄに入力されたテキストである。本実施形態では、Ｓ３３１４の判定は、Ｓ３３１３で受信したアクション要求の中に入力項目の値が含まれていたか否かの判定であるものとする。入力項目の値を受信した場合（アクション要求に入力項目の値が含まれていた場合）にはＳ３３１５に進み、そうでない場合にはＳ３３１６へ進む。

【０４０５】
Ｓ３３１５では、実行環境４００は、受信した入力項目の値を実行エンジンに含まれるメモリに一時記憶する。

【０４０６】
Ｓ３３１６では、実行環境４００は、実行環境用プログラム３４２４を実行することにより、要求されたアクションを実行する。入力項目の値を受信していた場合には、入力項目の値も用いてアクションを実行する。例えば、入力項目の値を引数として実行環境用プログラム３４２４のうち、要求されたアクションの部分を実行する。

【０４０７】
Ｓ３３１７では、実行環境４００は、Ｓ３３１６でのアクションの実行の結果、アプリユーザー用端末２００に表示されるアプリの画面に表示すべき値である、出力項目の値があるか否かを判定する。例えば、アクションが“四則演算を行う”といったアクションであった場合には、演算の解が出力項目の値として得られる。また、例えばアクションが“画面遷移する”や、“データベースに記録する”といったアクションであった場合には、アク

ションの結果としての出力項目の値は無いこともある。出力項目の値がある場合にはＳ３３１８に進み、そうでない場合にはＳ３３１９へ進む。

【０４０８】

Ｓ３３１８では、実行環境４００は、Ｓ３３１６で実行したアクションの結果情報として、出力項目の値を含むアクション結果情報を生成する。

【０４０９】

Ｓ３３１９では、実行環境４００は、Ｓ３３１６で実行したアクションの結果情報をアプリユーザー用端末２００に送信する。Ｓ３３１８で出力項目の値を含むアクション結果情報が生成されていた場合には、出力項目の値もアプリユーザー用端末２００に送信されることになる。そして、後述する図３３（ｃ）のＳ３３４８で、アプリユーザー用端末２００の画面に出力項目の値が表示される。アクションの結果情報には、画面遷移の指示が含まれることもある。画面遷移の指示が含まれていた場合には、後述する図３３（ｃ）のＳ３３４７の処理によって、アプリユーザー用端末２００のディスプレイ１０５に表示されるアプリの画面において画面遷移が発生する。

【０４１０】

Ｓ３３２０では、実行環境４００は、処理を終了するか否かを判定する。実行環境４００は、処理を終了すると判定した場合（Ｓ３３２０でＹｅｓ）、処理を終了する。実行環境４００は、処理を終了しないと判定した場合（Ｓ３３２０でＮｏ）、処理はＳ３３１３に戻る。例えば、Ｓ３３５０で後述するアプリを終了させるイベントが発生した場合に処理を終了すると判定する。

図３３（ｃ）のＳ３３３１では、アプリユーザー用端末２００は（アプリユーザー用端末２０１を含むが、以下ではアプリユーザー用端末２００を例として説明する）、インターネットブラウザソフトを用いて、実行環境４００にデプロイ済み（構築済み、生成済み）のアプリケーションに対してアクセスを行う。より具体的には、デプロイ済みのアプリのＵＲＬ（実行環境にアクセスするＵＲＬ）を指定してアクセスする操作（例えばアプリのＵＲＬのリンクをクリックしたり、アドレスバーにアプリのＵＲＬを入力してＥｎｔｅｒキーを押下する操作）があったことに応じて、アプリのＵＲＬにアクセス（接続）する。

【０４１１】

Ｓ３３３２では、アプリユーザー用端末２００は、クライアント用プログラムを受信したか否かを判定する。実行環境４００は、アプリユーザー用端末２００からのアクセスを検知すると、実行環境の配信エンジン（配信エンジン４１５、４５５、４６５、４７５などのうちアクセスされた実行環境のもの）が、ストレージに記録されたクライアント用プログラム（クライアント用プログラム４２２、４５６ｃ、４６６ｃ、４７６ｃなどのうち、アクセスされた実行環境のもの）をアプリユーザー用端末２００に送信する。Ｓ３３３２では、そのクライアント用プログラムを受信したか否かを判定する。クライアント用プログラムを受信した場合にはＳ３３３３に進み、そうでない場合にはＳ３３３２でライアント用プログラムの受信を待つ。

【０４１２】

Ｓ３３３３では、アプリユーザー用端末２００は、実行環境４００にＵＩ定義情報の取得要求を送信する。Ｓ３３３２で受信したクライアント用プログラムに、実行環境にアクセスしたらまずはＵＩ定義情報の取得要求を出す旨が定義されている。Ｓ３３３３はそれに従った処理である。ここで送信したＵＩ定義情報取得要求が、前述した図３３（ｂ）のＳ３３１１において実行環境４００に受信される。

【０４１３】

Ｓ３３３４では、アプリユーザー用端末２００は、実行環境４００からＵＩ定義情報を受信したか否かを判定する。ここで受信されるＵＩ定義情報は前述した図３３（ｂ）のＳ３３１２で実行環境４００から送信されるものである。ＵＩ定義情報を受信した場合は、ＵＩ定義情報をメモリ１０２に記録してＳ３３３５に進み、そうでない場合にはＳ３３３４でＵＩ定義情報の受信を待つ。ＵＩ定義情報はメモリ１０２（ワークメモリ）に一時的な情報として記録され、アプリを終了したことに応じて（アプリのＵＲＬへの接続終了に

応じて）、自動的に消去するものとする。また、アプリユーザー用端末２００は、メモリ１０２に記録したＵＩ定義情報に基づいてアプリの画面をディスプレイ１０５に表示する。

【０４１４】

Ｓ３３３５では、アプリユーザー用端末２００は、アクションのトリガーが発生したか否かを判定する。具体的には、アプリの画面において、画面遷移を指示する操作（遷移先のキャンバスのアクションの起動トリガー）、アプリの画面に表示されたＵＩ部品に対する操作（クリック等によるＵＩ部品のアクションのトリガー）があったか否かを判定する。アクションのトリガーがあった場合にはＳ３３３６へ進み、そうでない場合にはＳ３３５０へ進む。

【０４１５】

Ｓ３３３６では、アプリユーザー用端末２００は、メモリ１０２に記録されたＵＩ定義情報（図３４におけるＵＩ定義情報３４３１）のうち、Ｓ３３３５で検出したトリガーに応じて実行すべきアクションに関する記述部分を抽出する。この記述部分は、ＪａｖａＳｃｒｉｐｔ（プログラミング言語）で記載されたテキスト（文字列）の情報である。

【０４１６】

Ｓ３３３７では、アプリユーザー用端末２００は、Ｓ３３３６で抽出した記述部分から、半角の「＄ｕｉ」という文字列を先頭から検索する。

【０４１７】

Ｓ３３３８では、アプリユーザー用端末２００は、Ｓ３３３７の検索の結果、「＄ｕｉ」という文字列があったか否かを判定する。「＄ｕｉ」があった場合にはＳ３３３９に進み、そうでない場合にはＳ３３４３に進む。

【０４１８】

Ｓ３３３９には、アプリユーザー用端末２００は、Ｓ３３３８で見つかった「＄ｕｉ」の文字列が、当該文字を含む一文の中で左辺にあるか否かを判定する。「一文」（１つの文）は、プログラミング言語での文字列において行終端記号（例えばセミコロン「；」）や、閉じ括弧「｝」で区切られるまでの文字列である。すなわち、２つの文は行終端記号または閉じ括弧（'｝'）で区切られる。検索で見つかった「＄ｕｉ」が、一文の中で、「＝」（等号、イコール）よりも左側に位置していれば、左辺にあると判定される。「＄ｕｉ」が左辺にある場合にはＳ３３４０に進み、そうでない場合にはＳ３３４１に進む。

【０４１９】

Ｓ３３４０では、アプリユーザー用端末２００は、Ｓ３３３８で見つかった「＄ｕｉ」で始まる所定の構造の要素文字列（＄ｕｉ．ＵＩ部品ＩＤ．対象となる情報の種別）に基づく情報を、メモリ１０２に生成した項目定義リスト（図３４の項目定義リスト３４３３）に出力項目として記録する。具体的には、「＄ｕｉ」で始まる「．」（ピリオド）で区切られた構造の要素文字列（＄ｕｉ．ＵＩ部品ＩＤ．対象となる情報の種別）からＵＩ部品ＩＤ（項目コード、コンポーネント識別子）を取得し、そのＵＩ部品が出力項目である旨を記録する。例えば、アクションＡに関するトリガーが発生し、アクションＡに関する記述部分から「＄ｕｉ」が見つかり、それが左辺にあり、ＵＩ部品ＩＤが「ＵＩ部品ＩＤ３」であった場合には、図３４の項目定義リスト３４３３のように、出力項目にＵＩ部品ＩＤ３がある旨が記録される。なお、項目定義リスト３４３３はワークメモリであるメモリ１０２に一時的に保持する情報であって、アクションの結果を受信して出力項目に反映するか（後述するＳ３３４８でＮｏまたはＳ３３４９の処理を終了）、アプリを終了したこと（Ｓ３３５０でＹｅｓ）に応じて、自動的に消去するものとする。

【０４２０】

Ｓ３３４１では、アプリユーザー用端末２００は、Ｓ３３３８で見つかった「＄ｕｉ」の文字列が、当該文字を含む一文の中で左辺以外にあるか否かを判定する。左辺以外にある場合とは、右辺にある場合（「＝」より右側にある）か、「＝」を含まない分にある場合である。左辺以外にある場合にはＳ３３４２へ進み、そうでない場合にはＳ３３３７へ進む。なお、Ｓ３３３９でＮｏと判定された場合は、「＄ｕｉ」の文字列が、当該文字を

含む一文の中で左辺以外にあることと同義であるため、Ｓ３３４１の判定をせずにＳ３３４２へ進むものとしてもよい。

【０４２１】

Ｓ３３４２では、アプリユーザー用端末２００は、Ｓ３３３８で見つかった「＄ｕｉ」で始まる所定の構造の要素文字列（＄ｕｉ．ＵＩ部品ＩＤ．対象となる情報の種別）に基づく情報を、メモリ１０２に生成した項目定義リスト（図３４の項目定義リスト３４３３）に入力項目として記録する。具体的には、「＄ｕｉ」で始まる所定の構造の要素文字列（＄ｕｉ．ＵＩ部品ＩＤ．対象となる情報の種別）からＵＩ部品ＩＤ（項目コード、コンポーネント識別子）を取得し、そのＵＩ部品が入力項目である旨を記録する。例えば、アクションＡに関するトリガーが発生し、アクションＡに関する記述部分から「＄ｕｉ」が見つかり、それが右にあり、ＵＩ部品ＩＤが「ＵＩ部品ＩＤ１」であった場合には、図３４の項目定義リスト３４３３のように、入力項目にＵＩ部品ＩＤ１がある旨が記録される。また、例えば、アクションＡに関するトリガーが発生し、アクションＡに関する記述部分から「＄ｕｉ」が見つかり、それが「＝」のない文にあり、ＵＩ部品ＩＤが「ＵＩ部品ＩＤ２」であった場合には、図３４の項目定義リスト３４３３のように、入力項目にＵＩ部品ＩＤ２がある旨が記録される。

【０４２２】

Ｓ３３４２の処理を終えると、Ｓ３３３７に進み、Ｓ３３３６で抽出した記述部分の続きの部分において、さらに「＄ｕｉ」の文字列を検索し、Ｓ３３３８で「＄ｕｉ」が無いと判定されるまでＳ３３３９～Ｓ３３４２の処理を繰り返す。すなわち、Ｓ３３３６で抽出した記述部分に含まれる全ての＄ｕｉの文字列について、Ｓ３３３９～Ｓ３３４２の処理を行い、入力項目であるか出力項目であるかを項目定義リスト３４３３に記録する。

【０４２３】

Ｓ３３４３では、アプリユーザー用端末２００は、Ｓ３３３５でアクションのトリガーがあったと判定した時点で、項目定義リスト３４３３に記録された入力項目の対象となる情報に、値があるかどうかを判定する。例えば、Ｓ３３３６で抽出した記述部分に「＄ｕｉ．ＵＩ部品ＩＤ１．ｖａｌｕｅ」と記載されていた場合、ＵＩ部品ＩＤ１のＵＩ部品内の「ｖａｌｕｅ」という種別の情報に値があるかどうかを判定する。ＵＩ部品ＩＤ１のＵＩ部品がＴｅｘｔＦｉｅｌｄであった場合には、ｖａｌｕｅは、そのＴｅｘｔＦｉｅｌｄに入力されたテキスト（文字列）である。すなわち、アプリの画面に表示されたＴｅｘｔＦｉｅｌｄであるＵＩ部品ＩＤ１に対して、アプリのユーザーがキーボードでテキストが入力されているかを判定し、テキストが入力されていればＹｅｓと判定し、空欄であればＮｏと判定する。この判定を、項目定義リスト３４３３に記録されたすべての入力項目について行う。入力項目の対象となる情報に、値がある場合にはＳ３３４４へ進み、そうでない場合にはＳ３３４５へ進む。

【０４２４】

Ｓ３３４４では、アプリユーザー用端末２００は、入力項目の値を含む、Ｓ３３３５で検知したトリガーに対応するアクションの実行を要求するアクション要求を生成する。例えば、入力項目の値として、アプリユーザー用端末２００のディスプレイ１０５に表示されるアプリの画面に表示されたＴｅｘｔＦｉｅｌｄであるＵＩ部品ＩＤ１に対して、アプリのユーザーがキーボードで入力したテキストの情報を含むアクション要求を生成する。

【０４２５】

Ｓ３３４５では、アプリユーザー用端末２００は、Ｓ３３３５で検知したトリガーに対応するアクションの実行を要求するアクション要求を実行環境４００に送信する。Ｓ３３４４で入力項目の値ありアクションが生成されていた場合には、入力項目の値も実行環境４００に送信される。例えば、入力項目の値として、アプリユーザー用端末２００のディスプレイ１０５に表示されるアプリの画面に表示されたＴｅｘｔＦｉｅｌｄであるＵＩ部品ＩＤ１に対して、アプリのユーザーがキーボードで入力したテキストの情報が実行環境４００に送信される。ここで送信したアクション要求が、実行環境４００で実行される前述した図３３（ｂ）のＳ３３１３で実行環境４００に受信され、Ｓ３３１４～Ｓ３３１９

で説明した通り、Ｓ３３３５で検知したトリガーに応じたアクションが実行環境４００で実行される。

【０４２６】

Ｓ３３４６では、アプリユーザー用端末２００は、Ｓ３３４５で送信したアクション要求に対応するアクションの結果を実行環境４００から受信したか否かを判定する。アクションの結果を受信した場合にはＳ３３４７へ進み、そうでない場合にはＳ３３４６でアクションの結果の受信を待つ。

【０４２７】

Ｓ３３４７では、アプリユーザー用端末２００は、Ｓ３３４６で受信したアクションの結果を反映してディスプレイ１０５に表示されたアプリの画面を更新する。この時もメモリ１０２に保持したＵＩ定義情報に基づいて表示を行う。例えば、アクションの結果に画面遷移の指示があった場合には、画面遷移を実行する。

【０４２８】

Ｓ３３４８では、アプリユーザー用端末２００は、Ｓ３３４６で受信したアクションの結果に、出力項目の値が含まれていたか否を判定する。出力項目の値が含まれていた場合にはＳ３３４９に進み、そうでない場合にはＳ３３５０へ進む。

【０４２９】

Ｓ３３４９では、アプリユーザー用端末２００は、Ｓ３３４６で受信したアクションの結果に含まれる出力項目の値を、メモリ１０２に保持した項目定義リスト３４３３に記録した出力項目の情報に基づいて、ディスプレイ１０５に表示される出力項目に表示する。これによって例えば、対象となる情報の種別がｖａｌｕｅであれば出力項目のＵＩ部品に出力項目の値となるテキストや数値が表示されたり、対象となる情報の種別がｃｏｌｏｒであれば出力項目のＵＩ部品の色が出力項目の値が示す色に変更されたりする。

【０４３０】

Ｓ３３５０では、アプリユーザー用端末２００は、アプリを終了させるイベントがあったか否かを判定する。アプリを終了させるイベントには、例えば、アプリのＵＲＬが示すアクセス先への接続の切断（インターネットブラウザを閉じる、アプリユーザー用端末２００の電源を切る、関係のない他のＵＲＬへの接続に変更する、など）がある。アプリを終了させるイベントが無い場合にはＳ３３３５に進み、アプリを終了させるイベントがあった場合には、アプリの画面の非表示として図３３（ｃ）の処理を終了する。

【０４３１】

図１０（ｄ）に示したアクションボード９１０に入力されたプログラミング言語での文字列を例にとって、Ｓ３３３８～Ｓ３３４２で説明した「＄ｕｉ」を含む文字列について入力項目と出力項目のいずれであるかを判別して仕分ける処理について、より具体的に説明する。

【０４３２】

２行目には「ｃｏｎｓｔ　ｕｓｅｒｉｄ　＝　＄ｕｉ．ｔｅｘｔ＿ｆｉｅｌｄ＿ａ．ｖａｌｕｅ；」と記載されている。この文において、「＄ｕｉ」が右辺にあるため、Ｓ３３４２の処理で、項目定義リストに「ｔｅｘｔ＿ｆｉｅｌｄ＿ａ」が入力項目として記録される。すなわち「ｕｓｅｒｉｄ」という変数の値に、アプリの画面において「ｔｅｘｔ＿ｆｉｅｌｄ＿ａ」にユーザーが入力した値が取得されて設定される。

【０４３３】

４行目の文において、「＄ｕｉ」を含む「＄ｕｉ．ｔｅｘｔ＿ａｒｅａ＿ａ．ｖａｌｕｅ」との文字列が左辺にあるため、Ｓ３３４０の処理で、項目定義リストに「ｔｅｘｔ＿ａｒｅａ＿ａ」が出力項目として記録される。すなわち「ｔｅｘｔ＿ａｒｅａ＿ａ」というＵＩ部品に、「＝」の右辺が示す値（アクションの結果得られる出力値）が表示されることとなる。この文は、「ｔｅｘｔ＿ａｒｅａ＿ａ」が出力項目であるという定義に加えて、その出力項目に「＝」の右辺が示す値（アクションの結果得られる出力値）を出力するという処理（ロジック）を含んでいる。このように、ロジック（入力項目と出力項目との少なくとも一方の定義とは異なる処理）を含む文の中において、入力項目と出力項目の

定義も行えるというのが本実施形態の特徴的な部分の１つである。６行目も同様に、ロジック部分であるとともに、「＄ｕｉ．ｔｅｘｔ＿ａｒｅａ＿ａ．ｖａｌｕｅ」が左辺にあるため出力項目として扱われる。

【０４３４】

また、以下に、「＝」の無い文に「＄ｕｉ」があるアクションの文字列の例を示す

【０４３５】

（各行先頭の数字は行数を示す補助文字であってプログラミング言語の本文ではない）

```
1　ＳＱＬＳａｖｅ（｛
2　　　ＰＲＥＦ＿ＣＯＤＥ：　＄ｕｉ．ＰＲＥＦ＿ＣＯＤＥ．ｖａｌｕｅ，
3　　　ＰＲＥＦ＿ＮＡＭＥ：　＄ｕｉ．ＰＲＥＦ＿ＮＡＭＥ．ｖａｌｕｅ
4　｝）；
5　／／　画面遷移
6　＄ｆｎ．ｎｅｘｔＵＩ（'Ｌｉｓｔ＿ｉｐ＿ｄｅｍｏ'）；
```

上記のプログラミング言語での文字列は、アプリの画面の入力値（｀＄ｕｉ．ＰＲＥＦ＿ＣＯＤＥ．ｖａｌｕｅ｀，＄ｕｉ．ＰＲＥＦ＿ＮＡＭＥ．ｖａｌｕｅ）をＳＱＬＳａｖｅ関数のパラメータに使用してデータベースへ登録するアクションである。この場合、ＳＱＬＳａｖｅ関数のパラメータとして使用される＄ｕｉ．｛部品ＩＤ｝．｛対象となる情報の種別｝は（＝の左辺ではないため）入力項目を示すものとして識別される。

【０４３６】

また、「＄ｕｉ」を用いた所定の構造の要素文字列において、対象となる情報の種別は、ｖａｌｕｅ以外でもよく、例えば色でも良い。例えば、「＄ｕｉ．ＵＩ部品ＩＤ．ｃｏｌｏｒ　＝　＂＃００８０００＂；」と記述することで、ＵＩ部品の色を変更することができる。

【０４３７】

本実施形態では、アクションボードに入力するアクションの記述において、「＄ｕｉ」という特定の識別子が付された要素文字列の、文字列中の位置（より具体的には、１つの文の中における位置）に応じて、その要素文字列が示すＵＩ部品を入力項目して扱うか、出力項目として扱うかを、書き表せる（定義できる）ようにしている。従来であれば、入力項目とするか出力項目とするかはプログラミング言語でアクションの内容を入力する領域とは別の設定画面において行っていた。あるいは従来であれば、プログラミング言語でアクションの内容を入力する領域に記述するにしても、処理自体を示すロジック部分の文とは別の文で、いったん、入力項目して扱うか、出力項目として扱うかという変数を用いた宣言を行う必要があった。これに対し、本実施形態では、入力項目と出力項目の定義を、プログラミング言語でアクションロジックを記述した文字列の任意の記述箇所で定義することができる。そのため、ロジック記述部分の近辺で、そのロジックで用いる入力項目と出力項目を定義することができ、ロジックを記述する際に、そのロジックで用いる入力項目、出力項目を容易に設定することができる。また、本実施形態では、入力項目と出力項目に関数設定画面とアクションロジックの記述部分の間における値の受け渡しを、一度変数として定義する必要がない。そのため開発者が煩雑な変数管理をする必要が低減し、変数管理が容易となる。従ってアクションを記述する際のミスを生む要因が減り、ミスを低減することができる。このように、本実施形態では、プログラミング言語で入力されるロジックに用いる入力項目または／及び出力項目をより容易に設定することが可能である。

【０４３８】

また、「＄ｕｉ」という特定の識別子が付された要素文字列の位置によって入力項目／出力項目のいずれとするかを書き表す手法は一般的ではないため、通常のＪａｖａＳｃｒｉｐｔのプログラミング言語に基づくプログラム実行では、入力項目／出力項目のいずれとするかが上手く解釈されず、意図通り動作しない。そこで本実施形態では、Ｓ３３３８～Ｓ３３４２で説明した「＄ｕｉ」を含む文字列について入力項目と出力項目のいずれであるかを特定して仕分ける処理を行うことで、正しく動作する仕組みとしている。

【０４３９】
本実施形態では、特定の識別子を半角の「＄ｕｉ」とした例を説明したが、これに限るものではない。プログラミング言語において既存の他の識別子と混同されない文字列であれば他の文字列を識別子としてもよく、例えば、半角の「＿ｉｎｐｕｔｏｕｔｐｕｔ」としても良い。半角の「＄」はユニコードのドルである。半角の「＿」はユニコードのアンダーバーである。
【０４４０】
また、本実施形態では、図３３（ａ）のＳ３３０４で説明した通り、開発環境３００が、ＵＩ定義情報に基づいて実行環境４００で実行すべきプログラムである実行環境用プログラム３４１４を生成し、それを実行環境４００にデプロイする。アプリの実行時には、アプリユーザー用端末２００はＵＩ定義情報３４３１に基づいて処理を実行し、実行環境４００は実行環境用プログラム３４２３に基づいて処理を実行する。また、図３３（ｃ）のＳ３３３８～Ｓ３３４２で説明した通り、アプリユーザー用端末２００が、ＵＩ定義情報に基づいてアプリユーザー用端末２００で実行する処理で必要な項目定義リスト３４３３を生成する。このようにすることで、アプリの開発者は、構築済みアプリが実行される際に、実行環境４００で実行する処理に必要となる情報とアプリユーザー用端末２００で実行する処理に必要となる情報とを、開発者用端末１００で開発環境３００にアクセスしてアプリを開発する際には特段意識して区別する必要が無い。従って、開発するアプリケーションの実行の際に実行環境と端末装置とでそれぞれ用いる情報を開発者が明確に分別するように管理して開発する手間を低減し、ソフトウェア開発をより容易に行える。
【０４４１】
なお、本実施形態では、開発環境３００における保存処理の際に実行環境用プログラム３４１４を生成する例を説明したが、開発環境３００は実行環境用プログラム３４１４を生成せず、実行環境４００にＵＩ定義情報３４２１を記録した後に、実行環境４００がＵＩ定義情報３４２１に基づいて実行環境用プログラム３４２３を生成するようにしてもよい。
【０４４２】
＜アクション実行関連処理の変形例＞
図３５（ａ）に、定義情報の記録先となる開発環境３００が実行する保存処理（記録制御処理）のフローチャートの変形例を示す。この処理は、前述の図４のＳ４２２において、開発者用端末１００から送信された、定義情報を受信したことに連動して実行する処理である。
【０４４３】
また、図３６に、開発者用端末１００、開発環境３００、実行環境４００、アプリユーザー用端末２００、２０１に記録される情報の遷移の変形例を図示する。図３６は、図３５（ａ）～図３５（ｃ）のフローチャートの処理による情報の遷移の様子を模式的に表した図である。図３６において、図３４と同じ情報には同じ符号を付す。
【０４４４】
図３６では、アプリユーザー用端末２００においてＵＩ定義情報に基づいて項目定義リスト３４３３を生成する処理を行う代わりに、開発環境３００でＵＩ定義情報に基づいてクライアント用ＵＩ定義情報３６１５を生成する。そして、このクライアント用ＵＩ定義情報３６１５を実行環境４００にデプロイし（クライアント用ＵＩ定義情報３６２５として実行環境４００に記録される）、さらに実行環境４００からアプリユーザー用端末２００に送信する（クライアント用ＵＩ定義情報３６３５としてアプリユーザー用端末２００に記録される）。アクションの実行の際には、アプリユーザー用端末２００は、クライアント用ＵＩ定義情報３６３５に基づいて処理を行う。
【０４４５】
図３６では、図３４と異なり、アプリユーザー用端末２００にＵＩ定義情報を送信しない（アプリユーザー用端末２００に一時的にも記録させない）構成としている。このようにすることで、開発したアプリの詳細な定義がアプリユーザー用端末２００を介して漏洩

することを防止し、開発されたアプリの構成についての秘匿性を高めている。従って例えば、開発したアプリの少なくとも一部（例えばアクションボードに記述したＪａｖａＳｃｒｉｐｔのコード）を模倣したアプリなどが第３者によって作成される可能性を低減することができる。また、アプリユーザー用端末２００において、アクションのトリガーが発生するたびにＵＩ定義情報を解析して項目定義リスト３４３３を生成する処理（図３３（ｃ）のＳ３３３８～Ｓ３３４２の処理）を行わなくてよい。その分、アプリユーザー用端末２００における処理負荷を低減し、快適なレスポンス（応答速度）のアプリ動作を実現することができる。

【０４４６】

図３５（ａ）のＳ３５０１～Ｓ３５０４は、前述した図３３（ａ）のＳ３３０１～３３０４と同じ処理であるため説明を省略する。図３５（ａ）では図３３（ａ）の処理に加えて、Ｓ３５０５以降の処理を行う。

【０４４７】

Ｓ３５０５では、開発環境３００は、クライアント用ＵＩ定義情報３６１５を生成する。クライアント用ＵＩ定義情報３６１５は例えば「ｕｉＤｅｆ２．ｊｓｏｎ」というファイル名のＪｓｏｎ形式のファイルであるものとする。クライアント用ＵＩ定義情報３６１５には、ＵＩ定義情報３４１１から取得した情報として、イニシャルＵＩ等のアプリに関する各設定情報、ＵＩ画面毎のＵＩ部品（コンポーネント）の情報（ＵＩ部品ＩＤやプロパティで設定された内容、ＵＩ部品の配置位置やサイズなど）、キャンバスのアクションの識別子、ＵＩ部品のアクションの識別子などが含まれる。また、クライアント用ＵＩ定義情報３６１５には、アクション毎の入出力項目定義（入力項目、出力項目にそれぞれどの部品があるかという定義）も記録される。Ｓ３５０５の時点ではアクション毎の入出力項目定義の中身は未挿入である（後述するＳ３５０６～３５１２の処理で挿入される）。また、アクションボードに記載されたＪａｖａＳｃｒｉｐｔ（プログラミング言語）で記述された文字列（アクションのソースコード）自体は記録されないものとする。すなわち、生成されるクライアント用ＵＩ定義情報３６１５は、アクションの識別子やアクション毎の入出力項目定義の情報を含むが、アクションの動作内容を示す情報は含まない。

【０４４８】

Ｓ３５０６では、開発環境３００は、ＵＩ定義情報３４１１のうち、アクション記述部分３４１２を抽出する。この処理では、特定のアクションに関する記述だけではなく、全てのアクションに関する記述部分を抽出する。

【０４４９】

Ｓ３５０７では、開発環境３００は、Ｓ３５０６で抽出したアクション記述部分について、「＄ＵＩ」の文字列を先頭から検索する。

【０４５０】

Ｓ３５０８では、開発環境３００は、Ｓ３３３７の検索の結果、「＄ｕｉ」という文字列があったか否かを判定する。「＄ｕｉ」があった場合にはＳ３５０９に進み、そうでない場合には処理を終了する。Ｓ３５０８でＮｏとなるのは、全ての「＄ｕｉ」の検索が終わったか、アクション記述部分に全く「＄ｕｉ」が無かった場合である。

【０４５１】

Ｓ３５０９では、開発環境３００は、図３３（ｃ）のＳ３３３９と同様、Ｓ３５０８で見つかった「＄ｕｉ」の文字列が、当該文字を含む一文の中で左辺にあるか否かを判定する。「＄ｕｉ」が左辺にある場合にはＳ３５１０に進み、そうでない場合にはＳ３５１１に進む。

【０４５２】

Ｓ３５１０では、開発環境３００は、Ｓ３３３８で見つかった「＄ｕｉ」で始まる所定の構造の要素文字列（＄ｕｉ．ＵＩ部品ＩＤ．対象となる情報の種別）に基づく情報を、Ｓ３５０５で生成したクライアント用ＵＩ定義情報３６１５に出力項目として記録する。より詳しくは、クライアント用ＵＩ定義情報３６１５のうち、出力項目を定義する配列に、Ｓ３３３８で見つかった「＄ｕｉ」で始まる所定の構造の要素文字列（＄ｕｉ．ＵＩ部

品ＩＤ．対象となる情報の種別）からＵＩ部品ＩＤを取得して、アクション毎の入出力項目定義のうち、“ｕｉＩｔｅｍｓＯｕｔ”の配列に記録する。例えばＵＩ部品ＩＤが「ＲＥＳＵＬＴ」であることを示す「＄ｕｉ」が左辺に見つかった場合、図３６に図示したように記録される。

【０４５３】

Ｓ３５１１では、開発環境３００は、図３３（ｃ）のＳ３３４１と同様、Ｓ３５０８で見つかった「＄ｕｉ」の文字列が、当該文字を含む一文の中で左辺以外にあるか否かを判定する。左辺以外にある場合にはＳ３５１２へ進み、そうでない場合にはＳ３５０７へ進む。なお、Ｓ３５０９でＮｏと判定された場合は、「＄ｕｉ」の文字列が、当該文字を含む一文の中で左辺以外にあることと同義であるため、Ｓ３５１１の判定をせずにＳ３５１２へ進むものとしてもよい。

【０４５４】

Ｓ３５１２では、開発環境３００は、Ｓ３３３８で見つかった「＄ｕｉ」で始まる所定の構造の要素文字列（＄ｕｉ．ＵＩ部品ＩＤ．対象となる情報の種別）に基づく情報を、Ｓ３５０５で生成したクライアント用ＵＩ定義情報３６１５に入力項目として記録する。より詳しくは、クライアント用ＵＩ定義情報３６１５のうち、入力項目を定義する配列に、Ｓ３３３８で見つかった「＄ｕｉ」で始まる所定の構造の要素文字列（＄ｕｉ．ＵＩ部品ＩＤ．対象となる情報の種別）からＵＩ部品ＩＤを取得して、アクション毎の入出力項目定義のうち、“ｕｉＩｔｅｍｓＩｎ”の配列に記録する。例えばＵＩ部品ＩＤが「ＩＮＰＵＴ＿１」であることを示す「＄ｕｉ」が右辺に見つかった場合、図３６に図示したように記録される。また、例えばＵＩ部品ＩＤが「ＩＮＰＵＴ＿２」であることを示す「＄ｕｉ」が「＝」のない文に見つかった場合、図３６に図示したように記録される。

【０４５５】

Ｓ３５１２の処理を終えると、Ｓ３５０７に進み、Ｓ３５０６で抽出した記述部分の続きの部分において、さらに「＄ｕｉ」の文字列を検索し、Ｓ３５０８で「＄ｕｉ」が無いと判定されるまでＳ３５０７～Ｓ３５１２の処理を繰り返す。すなわち、Ｓ３５０６で抽出した記述部分に含まれる全ての＄ｕｉの文字列について、Ｓ３５０９～Ｓ３５１２の処理を行い、入力項目であるか出力項目であるかをクライアント用ＵＩ定義情報３６１５に記録する。

【０４５６】

図３５（ｂ）に、実行環境４００におけるアプリ実行処理の変形例のフローチャートを示す。この処理は、実行環境４００の実行エンジンが実行する処理であり、アプリユーザー用端末２００または２０１から、デプロイ済み（構築済み、生成済み）のアプリケーションに対するアクセスがあった場合に実行する処理である。

【０４５７】

図３３（ｃ）に、アプリユーザー用端末２００または２０１おけるアプリ実行処理の変形例のフローチャートを示す。この処理は、アプリユーザー用端末２００または２０１のＣＰＵ１０１が実行する処理であり、アプリユーザー用端末２００または２０１のブラウザソフトで、実行環境４００にデプロイ済み（構築済み、生成済み）のアプリケーションに対してアクセスした場合に実行する処理である。また、図３５（ｂ）の実行環境４００におけるアプリ実行処理と、図３５（ｃ）のアプリユーザー用端末２００または２０１におけるアプリ実行処理とは、連動して行われる処理である。以下、アプリユーザー用端末２００または２０１による処理について、代表してアプリユーザー用端末２００が実行するものとして説明する（アプリユーザー用端末２０１に関する説明は同様であるため省略する）。

【０４５８】

図３５（ｂ）のＳ３５２１では、実行環境４００は、アプリユーザー用端末２００から送信されたＵＩ定義情報の取得要求を受信したか否かを判定する。ここで受信するＵＩ定義情報の取得要求は、図３５（ｃ）のＳ３５４３でアプリユーザー用端末２００から送信されるものである。ＵＩ定義情報の取得要求を受信した場合はＳ３５２２に進み、そうで

ない場合はＳ３５２１で待機する。
【０４５９】
　Ｓ３５２２では、実行環境４００は、ＵＩ定義情報３４１１をアプリユーザー用端末２００に送信することなく、クライアント用ＵＩ定義情報３６１５を実行環境４００に送信する。これによって、図３６に図示した通り、実行環境４００に記録されたクライアント用ＵＩ定義情報３６２５がアプリユーザー用端末２００にダウンロードされ、クライアント用ＵＩ定義情報３６３５として記録される。クライアント用ＵＩ定義情報３６１５とクライアント用ＵＩ定義情報３６３５とは同じ情報である。
【０４６０】
　Ｓ３５２３～Ｓ３５３０の処理は、前述した図３３（ｂ）のＳ３３１３～Ｓ３３２０とそれぞれ同様の処理であるため説明を省略する。
【０４６１】
　図３５（ｃ）のＳ３５４１～Ｓ３５４３は、前述した図３３（ｃ）のＳ３３３１～Ｓ３３３３とそれぞれ同様の処理であるため説明を省略する。ただし、Ｓ３５４３では、クライアント用ＵＩ定義情報の取得要求を送信する。
【０４６２】
　Ｓ３５４４は、アプリユーザー用端末２００は、実行環境４００からクライアント用ＵＩ定義情報を受信したか否かを判定する。ここで受信されるクライアント用ＵＩ定義情報は前述した図３５（ｂ）のＳ３５２２で実行環境４００から送信されるものである。クライアント用ＵＩ定義情報を受信した場合は、クライアント用ＵＩ定義情報をメモリ１０２に記録してＳ３５４５に進み、そうでない場合にはＳ３５４４でクライアント用ＵＩ定義情報の受信を待つ。クライアント用ＵＩ定義情報はメモリ１０２（ワークメモリ）に一時的な情報として記録され、アプリを終了したことに応じて（アプリのＵＲＬへの接続終了に応じて）、自動的に消去するものとする。また、アプリユーザー用端末２００は、メモリ１０２に記録したクライアント用ＵＩ定義情報３６３５に基づいてアプリの画面をディスプレイ１０５に表示する。
【０４６３】
　Ｓ３５４５では、アプリユーザー用端末２００は、前述した図３３（ｃ）のＳ３３３５と同様、アクションのトリガーが発生したか否かを判定する。アクションのトリガーがあった場合にはＳ３５４６へ進み、そうでない場合にはＳ３５５３へ進む。
【０４６４】
　Ｓ３５４６～Ｓ３５５３の処理は、前述した図３３（ｃ）のＳ３３４３～Ｓ３３５０とそれぞれ同様の処理であるため説明を省略する。前述した図３３（ｃ）のＳ３３３６～Ｓ３３４２に相当する処理は、変形例においてアプリユーザー用端末２００では行わない。
【０４６５】
　なお、上述の各フローチャートで説明した各種制御は１つのハードウェアが行ってもよいし、複数のハードウェア（例えば、複数のプロセッサーや回路）が処理を分担することで、装置全体の制御を行ってもよい。
また、本発明をその好適な実施形態に基づいて詳述してきたが、本発明はこれら特定の実施形態に限られるものではなく、この発明の要旨を逸脱しない範囲の様々な形態も本発明に含まれる。さらに、上述した各実施形態は本発明の一実施形態を示すものにすぎず、各実施形態を適宜組み合わせることも可能である
【０４６６】
（他の実施形態）
　本発明は、以下の処理を実行することによっても実現される。即ち、上述した実施形態の機能を実現するソフトウェア（プログラム）をネットワーク又は各種記憶媒体を介してシステム或いは装置に供給し、そのシステム或いは装置のコンピュータ（又はＣＰＵやＭＰＵ等）がプログラムコードを読み出して実行する処理である。この場合、そのプログラム、及び該プログラムを記憶した記憶媒体は本発明を構成することになる。
【０４６７】

＜複数の観点＞
上述の実施形態で説明した通り、本実施形態の情報処理システムは以下のような構成を含む。また、本実施形態は、以下の構成の各手段が実行するステップを有する情報処理システムの制御方法を示している。また、本実施形態は、以下の構成の各手段として少なくとも１つのコンピュータを機能させるためのプログラムによっても実現可能である。

【０４６８】
［構成１－１］
ユーザーがプログラミング言語で記述する入力とは異なる入力操作であって、創作関数を作成するための設定画面に対する前記ユーザーからの入力操作を受け付ける受付手段と、
前記受付手段で受け付けた前記入力操作に基づいて前記創作関数に関する情報を設定する設定手段と、
プログラミング言語で記述された、前記創作関数の識別情報が含まれる文字列を取得した場合に、前記設定手段で設定された情報に基づいて、前記創作関数の機能を含む前記文字列で示される機能を実行可能とする制御を行う制御手段と、
を有することを特徴とする情報処理システム。

【０４６９】
［構成１－２］
ユーザーがプログラミング言語で記述する入力とは異なる入力操作であって、創作関数を作成するための設定画面に対する前記ユーザーからの入力操作に基づいて、前記創作関数に関する情報を設定する設定手段と、
プログラミング言語の記述領域に、前記創作関数の識別情報に一部一致する文字列が入力された場合に、前記創作関数の識別情報の選択肢を表示するように制御する表示制御手段と、
を有することを特徴とする情報処理システム。

【０４７０】
［構成１－３］
ユーザーがプログラミング言語で記述する入力とは異なる入力操作であって、創作関数を作成するための設定画面に対する前記ユーザーからの入力操作に基づいて、前記創作関数に関する情報を設定する設定手段と、
プログラミング言語の記述領域とともに、前記設定手段で設定された前記創作関数の識別情報を表示するように制御する表示制御手段と、
を有することを特徴とする情報処理システム。

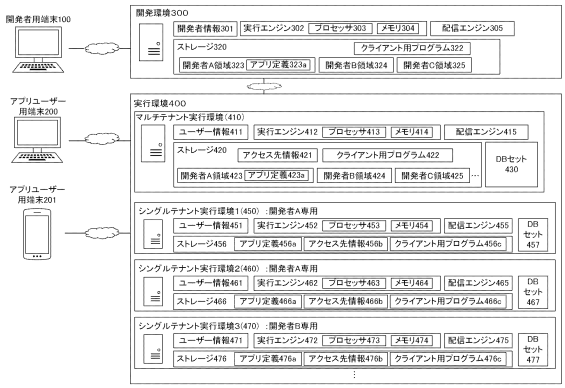
【０４７１】
［構成２］
アプリケーションの開発画面に配置された第１の種別のコンポーネントのうち指示領域（タブ）に対する操作があったことに応じて、前記第１の種別のコンポ－ントの領域内の少なくとも一部の領域の表示を、操作された指示領域に対応する要素画面に切り替えるように制御する表示制御手段と、
前記開発画面に配置された前記第１の種別のコンポーネントの領域に対して他のコンポーネントを配置する操作があった場合に、前記第１の種別のコンポーネントの複数の要素画面のうち、現在表示されている要素画面に対して前記他のコンポーネントを配置するように制御する制御手段と
を有することを特徴とする情報処理システム。

【０４７２】
［構成３］
アプリケーションの開発画面に第１の種別の第１のコンポーネントを配置する配置手段と、
前記配置手段で配置された前記第１のコンポ－ネントの設定値の入力領域に、デフォルトの設定値を、プログラミング言語で記述する際の構造を識別可能な形態で表示する
ように制御する表示制御手段と

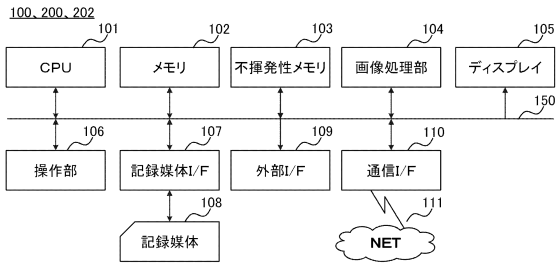
を有することを特徴とする情報処理システム。
【０４７３】
［構成４］
第１の種別の環境（開発環境）と第２の種別の環境（実行環境）とを含む情報処理システムであって、
前記第２の種別の環境であって、前記第１の種別の環境に接続しているユーザーがアクセス可能な環境のうち、特定の環境の特定の領域から、前記特定の環境におけるアクセス先情報を取得する取得手段と、
前記取得手段で取得したアクセス先情報が示すアクセス先に、所定の処理（例えばユーザー情報の送信処理）の実行要求を送信するように制御する送信制御手段と
を有することを特徴とする情報処理システム。
【０４７４】
［構成５］
第１の種別の環境（開発環境）と第２の種別の環境（実行環境）とを含む情報処理システムであって、
前記第１の種別の環境へアクセス可能なユーザーか否かのユーザー認証を行う認証手段と、
前記認証手段によって前記第１の種別の環境へアクセス可能と認証されたユーザーがアクセス可能な前記第２の種別の環境を特定する環境特定情報を取得する取得手段と、（アクセス可能実行環境リストの取得）
前記取得手段で取得した前記環境特定情報に基づき、前記第２の種別の環境に関する認証情報をユーザーから取得することなく、前記環境特定情報によって特定される環境へアクセスするように制御する制御手段と、
を有することを特徴とする情報処理システム。
【０４７５】
［構成６］
アプリケーションソフトウェアを構築するためのクライアント端末における開発画面として、構築される前記アプリケーションソフトウェアで表示される画面に配置するコンポーネントが操作された場合のアクションを入力する入力領域を表示するように制御する表示制御手段と、
前記入力領域に入力されたアクションの情報を含む前記アプリケーションソフトウェアの定義情報を記録するように制御する記録制御手段と、
前記アプリケーションソフトウェアの実行環境で実行すべきアクションを定義した第１の情報を、前記定義情報に基づいて生成するように制御する第１の制御手段と、
前記アプリケーションソフトウェアを実行する際に前記実行環境にアクセスする端末装置が用いる第２の情報を、前記定義情報に基づいて生成するように制御する第２の制御手段と、
を有することを特徴とする情報処理システム。
【０４７６】
［構成７］
表（データグリッド）の選択を受け付ける受付手段と、
選択された表の設定画面（プロパティボックス）を表示するように制御する表示制御手段と、
前記設定画面に対する第１の操作に応じて前記表に対してカラムを追加するように制御し、
前記設定画面に対する第２の操作に応じて、前記表の各カラムに対する設定を行うように制御する制御手段と
を有することを特徴とする情報処理装置。
【０４７７】
［構成８］

表の選択を受け付ける受付手段と、
選択された表に含まれる複数のカラム毎のアクションの入力画面にそれぞれ対応する複数の選択肢を表示するように制御する表示制御手段と、
前記複数の選択肢のうちいずれかが選択されたことに応じて、選択された選択肢に対応するカラムのアクションの入力画面を表示し、ユーザーからのアクションの入力操作を受け付けるように制御する制御手段と
を有することを特徴とする情報処理システム。
【０４７８】
［構成９］
プログラミング言語の入力領域に入力された文字列を取得する取得手段と、
前記取得手段で取得した文字列のうち、特定の識別子が付された要素文字列に基づき、当該要素文字列が、入力として取得すべき情報である入力項目と、出力すべき情報である出力項目とのいずれかの情報であることを特定する特定手段と、
前記特定手段で特定された内容を反映して、前記文字列に基づくプログラムを実行するように制御する制御手段と、
を有することを特徴とする情報処理システム。
【０４７９】
［構成１０］
プログラミング言語でソースコードを記述する操作を含まず、選択肢から選択する操作を含む操作群に基づいて、構築されるアプリケーションソフトウェアの画面に対してアクションを定義する、あるいは構築される前記アプリケーションソフトウェアの画面に表示されるコンポーネントとしてアクションが定義された自動生成部品を配置するように制御する制御手段と（例えば、ワークフロー生成によるアクションが定義されたボタンの配置、キャンバスにアクションが定義されたＵＩ画面、ＣＲＵＤボタン配置によるボタンの配置、ＵＩ画面の生成、等）、
前記制御手段によってアクションが定義された画面と前記自動生成部品とのいずれかが選択され、アクションを表示する指示があったことに応じて、前記選択がなされた対象に定義されたアクションをプログラミング言語での文字列で表示するように制御する表示制御手段と、
前記表示制御手段によって表示された、アクションを表すプログラミング言語での前記文字列の修正操作を受け付ける受付手段と、
を有することを特徴とする情報処理システム。
【０４８０】
［構成１１］
アプリケーションの定義を行う操作を受け付ける受付手段と、
前記受付手段で受け付けた操作に基づいて定義された定義情報に基づいてネットワーク上の環境に前記アプリケーションをデプロイする指示を行う指示手段と、
前記指示手段による指示によって前記アプリケーションのデプロイが完了したことに応じて、前記アプリケーションを実行するためのアドレスであって、前記デプロイがされた場所を示すアクセス先情報を表示するように制御する表示制御手段と、
を有することを特徴とする情報処理システム。
【０４８１】
［構成１２］
構築するアプリケーションにおけるユーザー認証のための認証用画面のテンプレートとして、予め定められたアクションが定義された雛形コンポーネントが予め配置された前記認証用画面のレイアウト編集画面を表示するように制御する表示制御手段と、
前記レイアウト編集画面に前記雛形コンポーネントと異なる他のコンポーネントを配置する編集操作と、前記レイアウト編集画面おける前記雛形コンポ―ネントの表示形態を変更する編集操作と、のうち少なくともいずれかを受け付ける受付手段と、
前記受付手段で受け付けた編集操作を反映した前記認証用画面が表示されるアプリケーシ

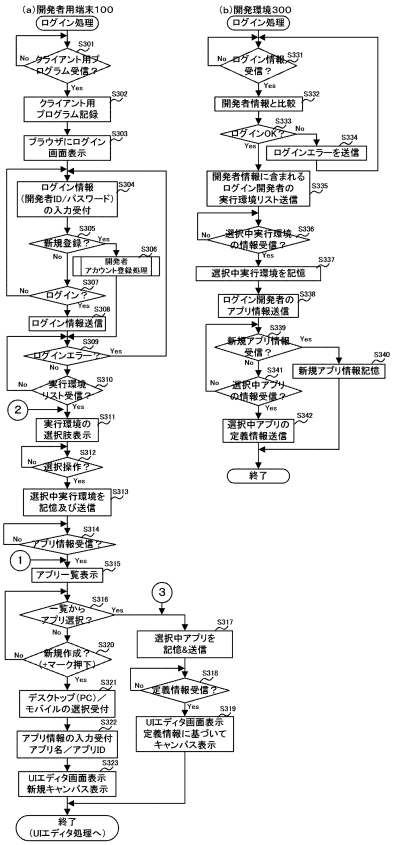
ョンを構築するための定義情報を記録するように制御する制御手段と、
を有することを特徴とする情報処理システム。
【０４８２】
［構成１３］
マルチテナント環境への新規アカウントの登録指示を受け付ける受付手段と、
前記受付手段で新規アカウントの登録指示を受け付けた場合に、
・前記マルチテナント環境に登録済みの有効なアカウント数が特定の閾値に達してない場合には、新規アカウントは第１のデータベース環境に接続し、
・前記マルチテナント環境に登録済みの有効なアカウント数が前記特定の閾値を超えている場合には、新規アカウントは前記第１のデータベース環境とは異なる第２のデータベース環境に接続する
ように制御する制御手段と、
を有することを特徴とする情報処理システム。
【０４８３】
発明は上記実施形態に制限されるものではなく、発明の精神及び範囲から離脱することなく、様々な変更及び変形が可能である。従って、発明の範囲を公にするために請求項を添付する。
【符号の説明】
【０４８４】
１００：開発者用端末、２００：アプリユーザー用端末、２０１：アプリユーザー用端末２０１、３００：開発環境、４００：実行環境
【図面】
【図１】



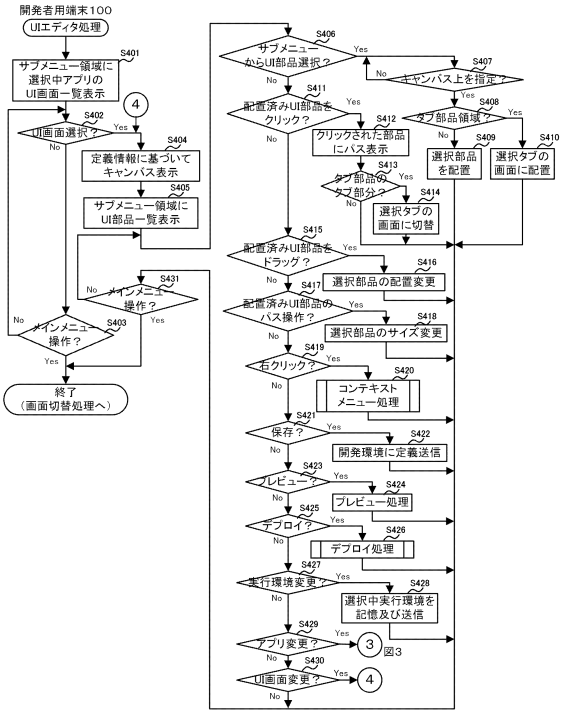
【図２】



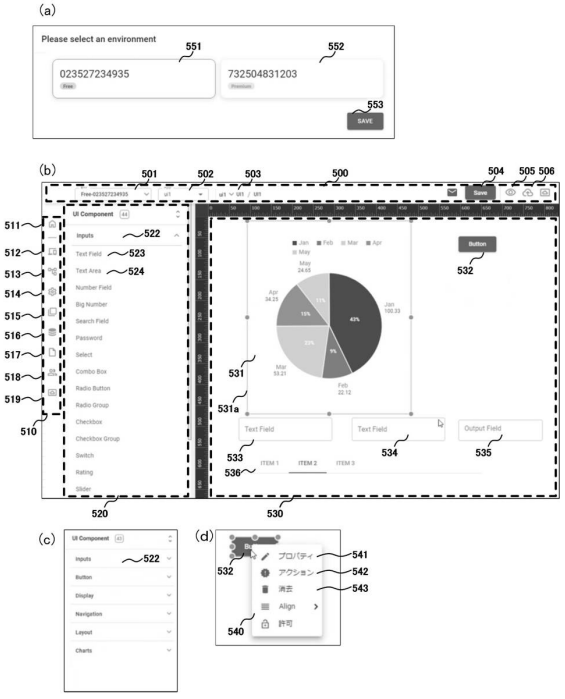
【図３】


(a)開発者用端末100
ログイン処理
S301
クライアント用プログラム受信？
S302
クライアント用プログラム記録
S303
ブラウザにログイン画面表示
S304
ログイン情報（開発者ID/パスワード）の入力受付
S305
新規登録？
S306
開発者アカウント登録処理
S307
ログイン？
S308
ログイン情報送信
S309
ログインエラー？
S310
実行環境リスト受信？
S311
実行環境の選択肢表示
S312
選択操作？
S313
選択中実行環境を記憶及び送信
S314
アプリ情報受信？
S315
アプリ一覧表示
S316
皆からアプリ選択？
S317
選択中アプリを記憶&送信
S318
定義情報受信？
S319
UIエディタ画面表示 定義情報に基づいてキャンバス表示
S320
新規作成？（×マーク押下）
S321
デスクトップ(PC)／モバイルの選択受付
S322
アプリ情報の入力受付 アプリ名／アプリID
S323
UIエディタ画面表示 新規キャンバス表示
終了（UIエディタ処理へ）
(b)開発環境300
ログイン処理
S331
ログイン情報受信？
S332
開発者情報と比較
S333
ログインOK？
S334
ログインエラーを送信
S335
開発者情報に含まれるログイン開発者の実行環境リスト送信
S336
選択中実行環境の情報受信？
S337
選択中実行環境を記憶
S338
ログイン開発者のアプリ情報送信
S339
新規アプリ情報受信？
S340
新規アプリ情報記憶
S341
選択中アプリの情報受信？
S342
選択中アプリの定義情報送信
終了

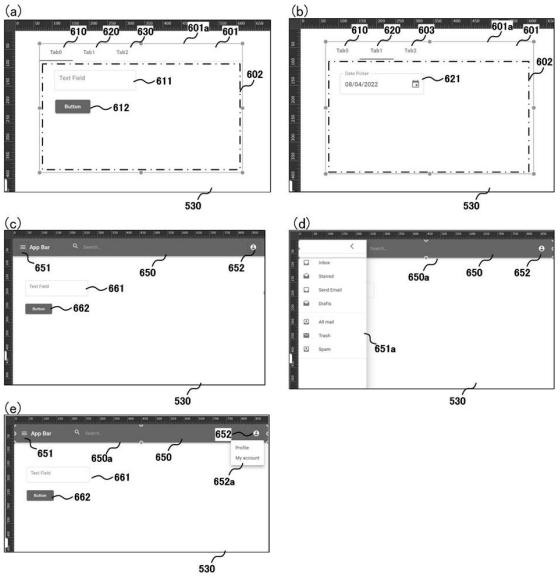
【図４】


開発者用端末100
UIエディタ処理
S401
サブメニュー領域に選択中アプリのUI画面一覧表示
S402
UI画面選択？
S403
メインメニュー操作？
S404
定義情報に基づいてキャンバス表示
S405
サブメニュー領域にUI部品一覧表示
S406
サブメニューからUI部品選択？
S407
キャンバス上を指定？
S408
タブ部品領域？
S409
選択部品を配置
S410
選択タブの画面に配置
S411
配置済みUI部品をクリック？
S412
クリックされた部品にパス表示
S413
タブ部品？
S414
選択タブの画面に切替
S415
配置済みUI部品をドラッグ？
S416
選択部品の配置変更
S417
配置済みUI部品のパス操作？
S418
選択部品のサイズ変更
S419
右クリック？
S420
コンテキストメニュー処理
S421
保存？
S422
開発環境に定義送信
S423
プレビュー？
S424
プレビュー処理
S425
デプロイ？
S426
デプロイ処理
S427
実行環境変更？
S428
選択中実行環境を記憶及び送信
S429
アプリ変更？
S430
UI画面変更？
S431
メインメニュー操作？
終了（画面切替処理へ）
図3

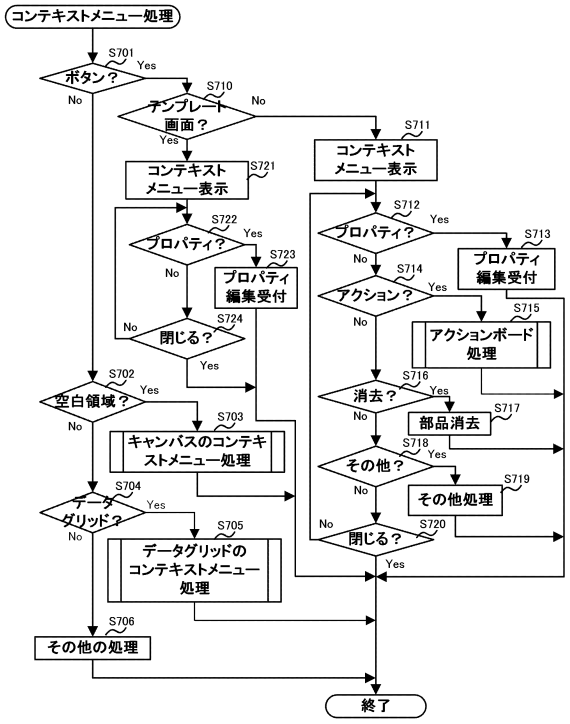
【図５】


(a)
Please select an environment
551
552
023527234935
732504831203
553
SAVE
(b)
501
502
503
500
504
505
506
511
512
513
514
515
516
517
518
519
510
520
530
522
523
524
532
531
531a
533
534
535
536
(c)
(d)
プロパティ
アクション
消去
Align
許可
541
542
543
540

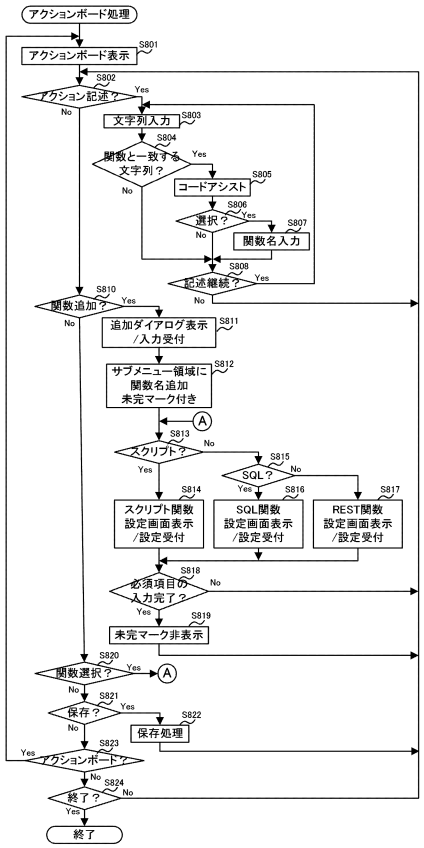
【図６】


(a)
610
620
630
601a
601
602
611
612
530
(b)
603
621
(c)
650
652
651
661
662
(d)
650a
651a
(e)
652a

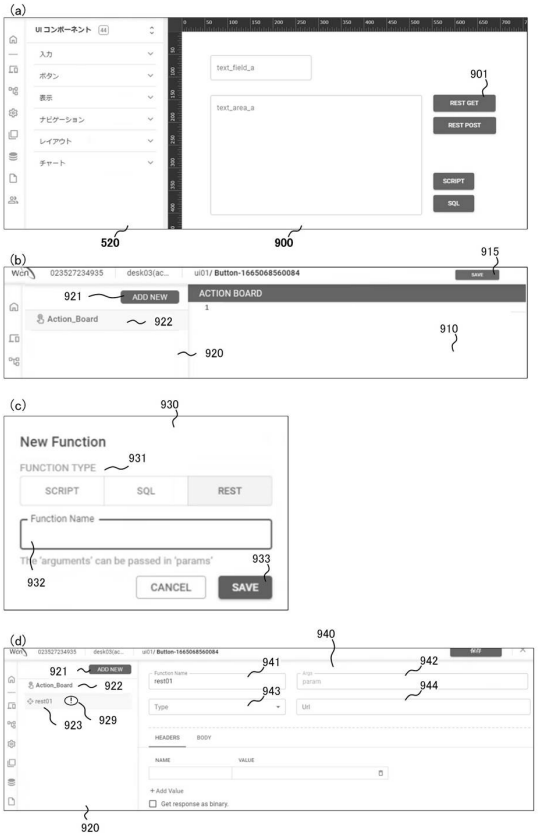
【図７】


コンテキストメニュー処理
S701
ボタン？
Yes
No
S710
テンプレート画面？
No
Yes
S711
コンテキストメニュー表示
S721
コンテキストメニュー表示
S712
プロパティ？
Yes
No
S713
プロパティ編集受付
S722
プロパティ？
Yes
No
S723
プロパティ編集受付
S714
アクション？
Yes
No
S715
アクションボード処理
S724
閉じる？
No
Yes
S716
消去？
Yes
No
S717
部品消去
S718
その他？
Yes
No
S719
その他処理
S720
閉じる？
No
Yes
S702
空白領域？
Yes
No
S703
キャンバスのコンテキストメニュー処理
S704
データグリッド？
Yes
No
S705
データグリッドのコンテキストメニュー処理
S706
その他の処理
終了

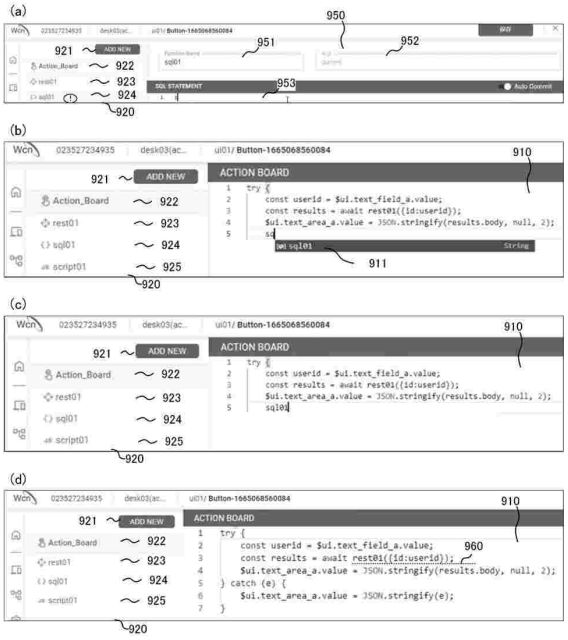
【図８】


アクションボード処理
S801
アクションボード表示
S802
アクション記述？
Yes
No
S803
文字列入力
S804
関数と一致する文字列？
Yes
No
S805
コードアシスト
S806
選択？
Yes
No
S807
関数名入力
S808
記述継続？
Yes
No
S810
関数追加？
Yes
No
S811
追加ダイアログ表示/入力受付
S812
サブメニュー領域に関数名追加未完マーク付き
A
S813
スクリプト？
No
Yes
S815
SQL？
No
Yes
S814
スクリプト関数設定画面表示/設定受付
S816
SQL関数設定画面表示/設定受付
S817
REST関数設定画面表示/設定受付
S818
必須項目の入力完了？
No
Yes
S819
未完マーク非表示
S820
関数選択？
Yes
No
A
S821
保存？
Yes
No
S822
保存処理
S823
アクションボード？
Yes
No
S824
終了？
No
Yes
終了

【図９】


(a)
UI コンポーネント
入力
ボタン
表示
ナビゲーション
レイアウト
チャート
text_field_a
text_area_a
REST GET
REST POST
SCRIPT
SQL
901
520
900
(b)
915
ADD NEW
ACTION BOARD
Action_Board
921
922
920
910
(c)
930
New Function
FUNCTION TYPE
931
SCRIPT
SQL
REST
Function Name
The 'arguments' can be passed in 'params'
932
CANCEL
SAVE
933
(d)
940
941
942
943
944
921
922
923
929
920

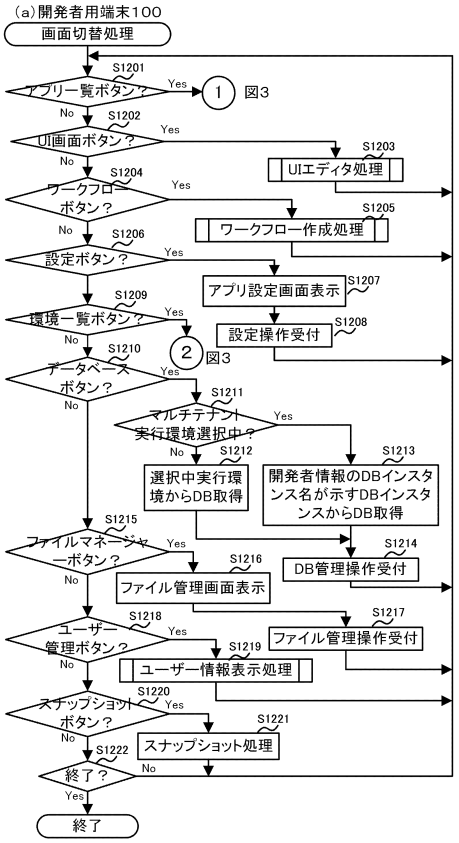
【図１０】


(a)
950
951
952
921
922
923
924
953
920
(b)
910
ADD NEW
ACTION BOARD
Action_Board
rest01
sql01
script01
921
922
923
924
925
920
try {
const userid = $ui.text_field_a.value;
const results = await rest01({id:userid});
$ui.text_area_a.value = JSON.stringify(results.body, null, 2);
sql01
911
(c)
910
ADD NEW
ACTION BOARD
sql01
920
(d)
910
ADD NEW
ACTION BOARD
960
} catch (e) {
$ui.text_area_a.value = JSON.stringify(e);
}
920

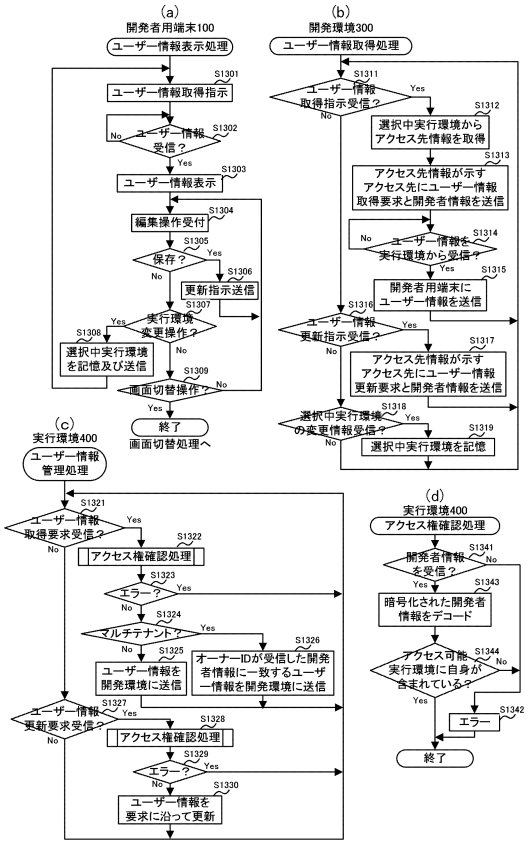
【図１１】

```
const rest01 = async (param) => {
  const toRestFnResult = (res) => {
    return {
      status: {
        code: res.status,
        text: res.statusText,
      },
      body: res.data,
      headers: res.headers,
    };
  };

    /** Parameters specified by dev user. */

  const defaultHeaders = {
  'Content-Type': 'application/json; charset=UTF-8',
  'Access-Control-Allow-Origin': '*',
  };


  let inputData;
const contentType = `none`;
  if (contentType === 'application/json' ||
    contentType === 'text/html' ||
    contentType === 'text/css' ||
    contentType   === 'text/plain' ||
    contentType === 'application/xml')   {
      inputData=``;
  }
  else {
    let data=`{}`;
    inputData = JSON.parse(data);
  }


  const method = `GET`;
  const url = `https://jsonplaceholder.typicode.com/users/${param.id}`;
  const binaryResponse = false;

  const headerParams = `{}`;

  const inputHeaders = JSON.parse(headerParams);

  let headers = { ...defaultHeaders, ...inputHeaders};

  let lambdaPayload = { method, url, data: inputData, headers: headers };
  if (method == 'GET') {
    delete lambdaPayload.data;
  }
  logger.log("LC>> lambdaPayload",lambdaPayload);

  if(binaryResponse){
    lambdaPayload.responseType='arraybuffer';
  }

  const client = new LambdaClient();
  const params = {
    FunctionName: 'wpcn-lambda-rest',
    InvocationType: 'RequestResponse',
    LogType: 'Tail',
    Payload: JSON.stringify(lambdaPayload),
  };

  const command = new InvokeCommand(params);
  let resultToString;
  try{
    let responseData = await client.send(command);
    resultToString = JSON.parse(Buffer.from(responseData.Payload));
    logger.log("LC>> result", resultToString);
  } catch (error) {
    logger.error("LC>> REST function failed: ", error);
    throw error;
  }
  return toRestFnResult(resultToString);
};
```

【図１２】



【図１３】



【図１４】

(a)開発者情報301(開発環境300に保持)

メールアドレス（ユーザーネーム）	パスワード	氏	名	アクセス可能な実行環境ID（クラウドサービスのアカウントID）	Verify	マルチテナント実行環境でのアクセス可能DBインスタンス名
aaaa@xxx.co.jp	aAaaa	A	a	マルチテナントの実行環境ID（１２桁），シングルテナント実行環境１のID（１２桁），シングルテナント実行環境２のID（１２桁）	confiemed	DBインスタンス１
bbbb@xxx.co.jp	bBbbb	B	b	マルチテナントの実行環境ID（１２桁），シングルテナント実行環境３のID（１２桁）	confiemed	DBインスタンス１
cccc@xxx.co.jp	cCccc	C	c	…	…	…
…	…	…	…	…	…	…

(b)ユーザー情報411（マルチテナント実行環境410に保持）

ユーザー別情報

アプリのユーザーネーム（メールアドレス）	パスワード	氏	名	Eメールの承認済み？	オーナーID（開発者ID）
aaaa@xxx.co.jp	aaAaa	A	a	true	aaaa@xxx.co.jp
bbbb@xxx.co.jp	bbBbb	B	b	true	aaaa@xxx.co.jp
cccc@xxx.co.jp	ccCcc	C	c	false	bbbb@xxx.co.jp
…	…	…	…	…	…

(c)ユーザー情報(シングルテナント実行環境に保持)

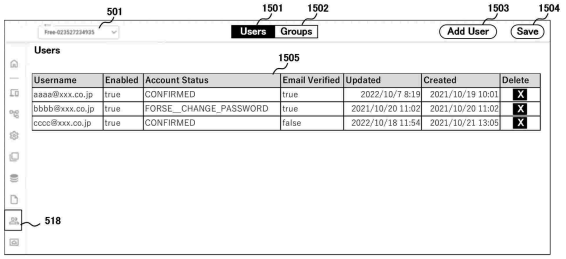
(c1)ユーザー別情報

アプリのユーザーネーム（メールアドレス）	パスワード	氏	名	Eメールの承認済み？
aaaa@xxx.co.jp	aaAaa	A	a	true
bbbb@xxx.co.jp	bbBbb	B	b	true
cccc@xxx.co.jp	ccCcc	C	c	false
…	…	…	…	…

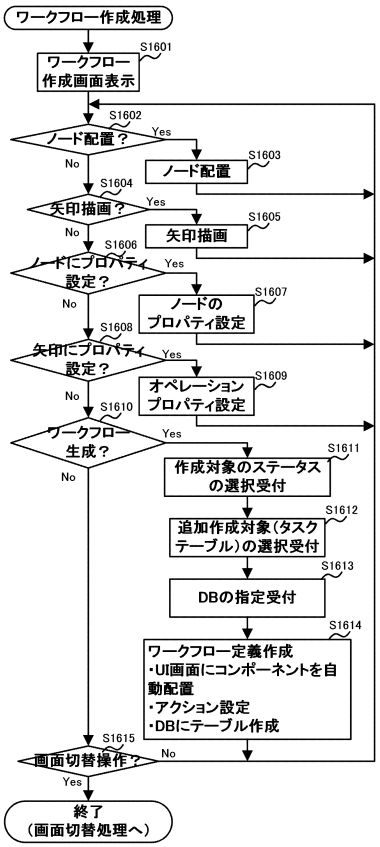
(c2)グループ情報

グループID	所属ユーザー
Agr	aaaa@xxx.co.jp; bbbb@xxx.co.jp
Bgr	cccc@xxx.co.jp
Cgr	aaaa@xxx.co.jp; cccc@xxx.co.jp
…	…

【図１５】



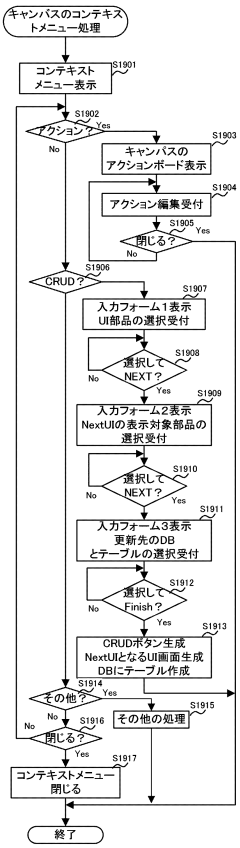
【図１６】



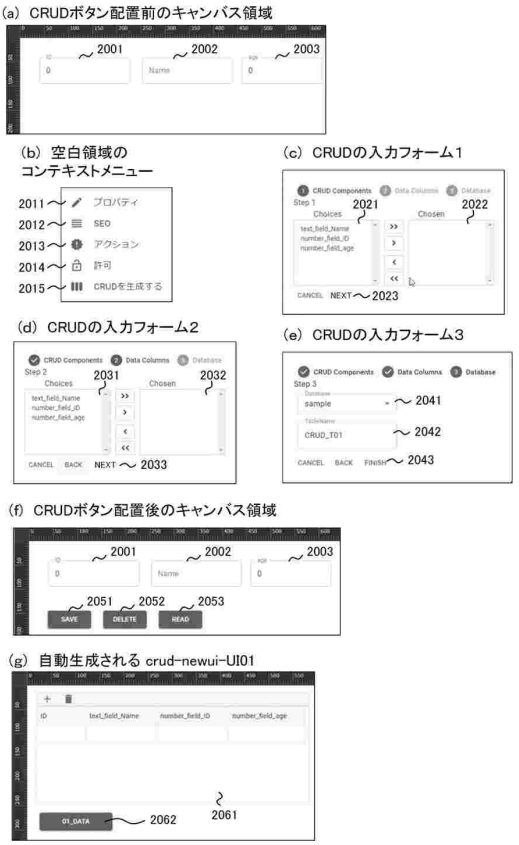
【図１７】

【図１８】

【図１９】

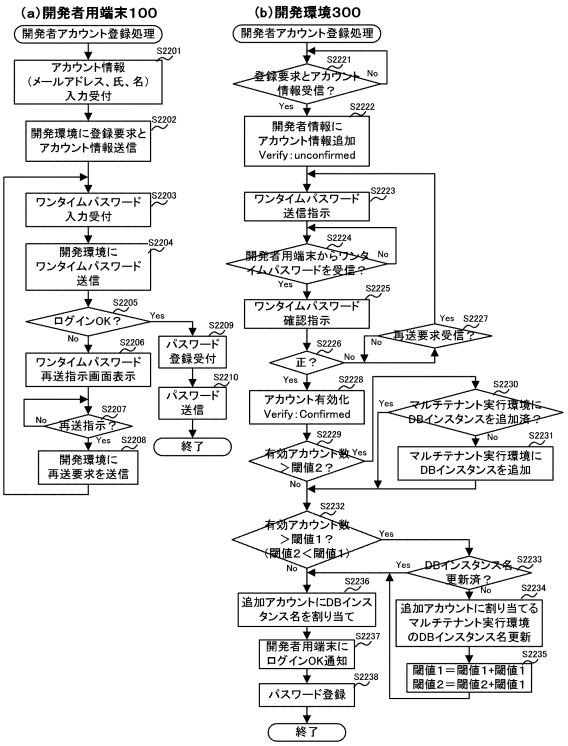


【図２０】



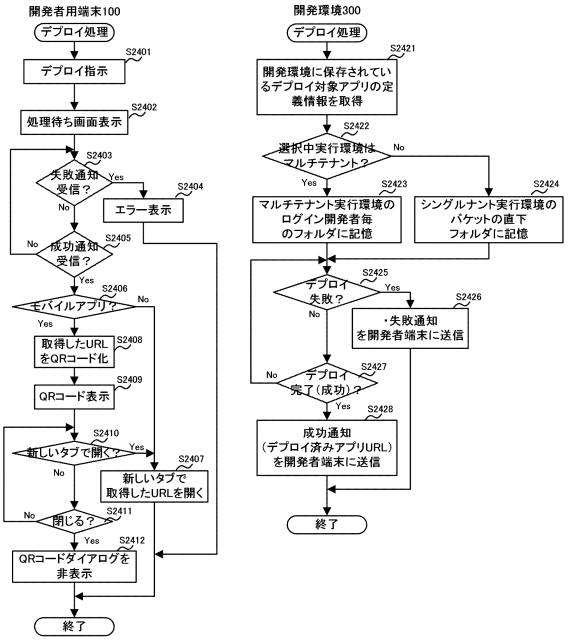
【図２１】

【図２２】

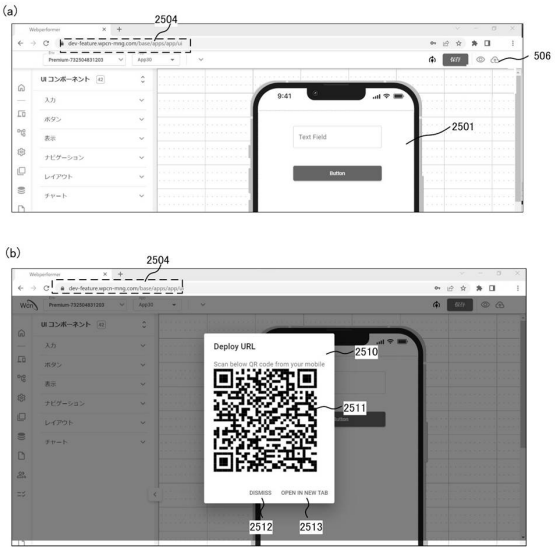


【図２３】

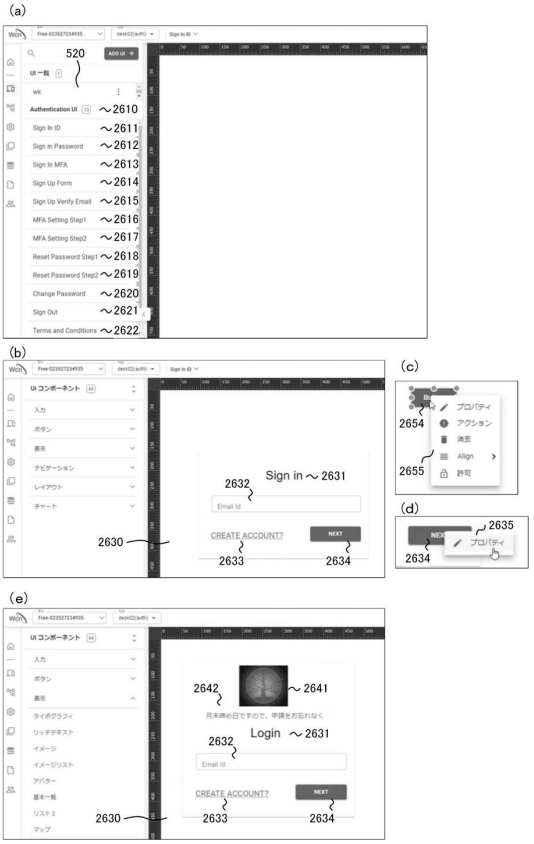
【図２４】



【図２５】

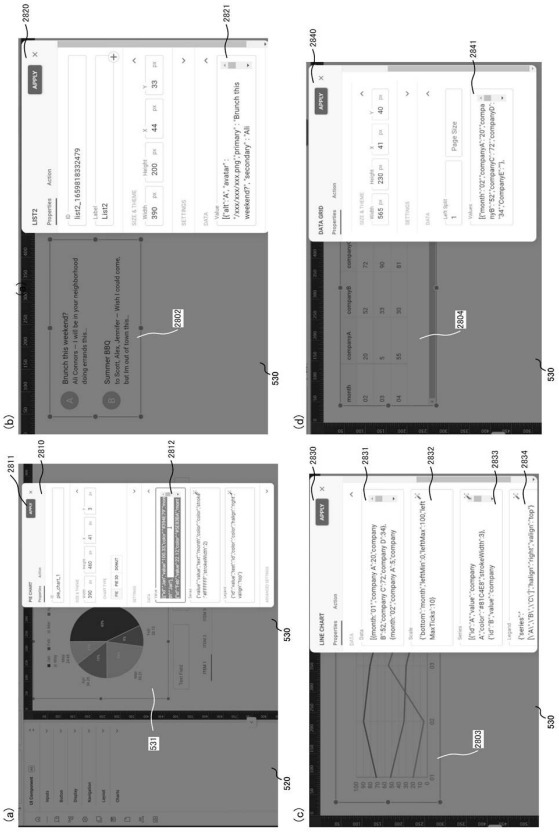


【図２６】

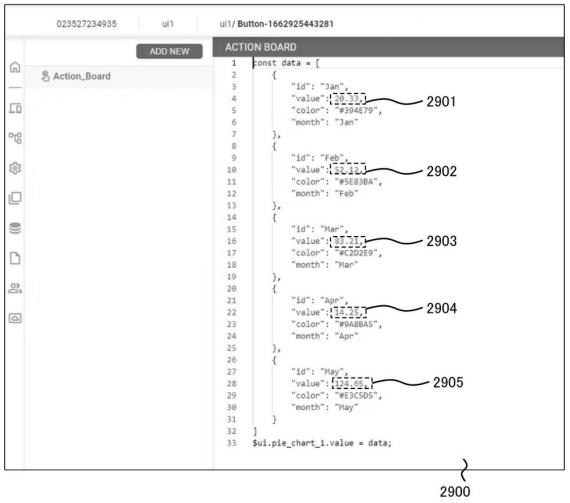


【図２７】

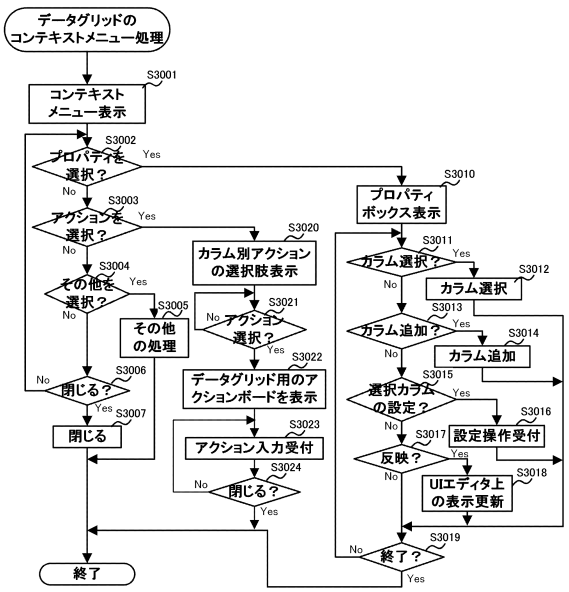
【図２８】



【図２９】

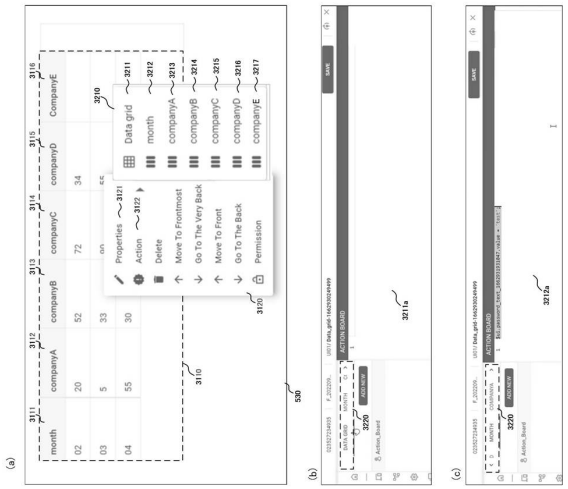


【図３０】

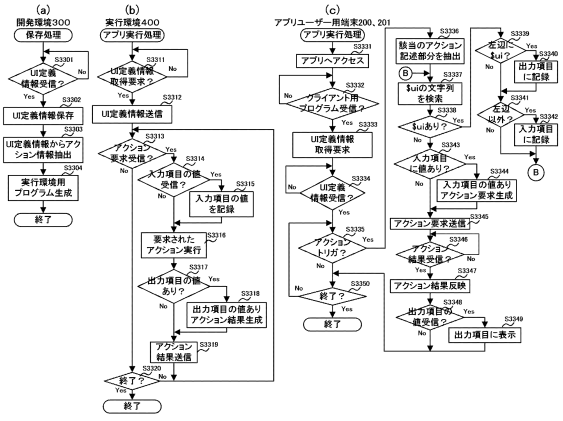


【図３１】

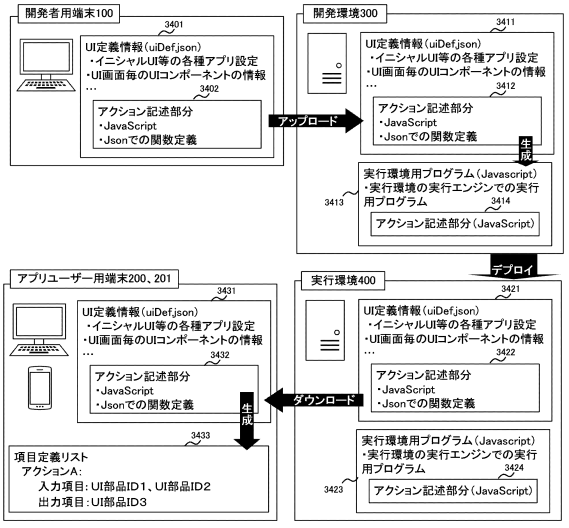
【図３２】



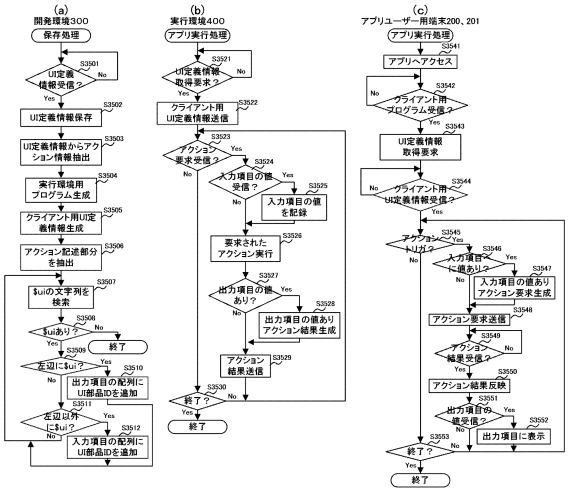
【図３３】



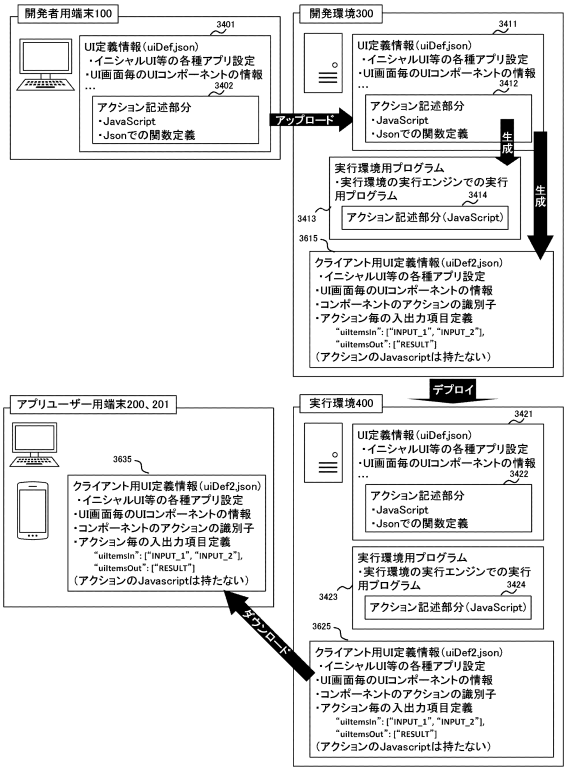
【図３４】



【図３５】


(a)
開発環境300
保存処理
S3501
UI定義情報受信？
No
Yes
S3502
UI定義情報保存
S3503
UI定義情報からアクション情報抽出
S3504
実行環境用プログラム生成
S3505
クライアント用UI定義情報生成
S3506
アクション記述部分を抽出
S3507
$uiの文字列を検索
S3508
$uiあり？
No
Yes
終了
S3509
左辺に$ui？
Yes
No
S3510
出力項目の配列にUI部品IDを追加
S3511
左辺以外に$ui？
Yes
No
S3512
入力項目の配列にUI部品IDを追加
(b)
実行環境400
アプリ実行処理
S3521
UI定義情報取得要求？
No
Yes
S3522
クライアント用UI定義情報送信
S3523
アクション要求受信？
Yes
No
S3524
入力項目の値受信？
Yes
No
S3525
入力項目の値を記録
S3526
要求されたアクション実行
S3527
出力項目の値あり？
Yes
No
S3528
出力項目の値ありアクション結果生成
S3529
アクション結果送信
S3530
終了？
No
Yes
終了
(c)
アプリユーザー用端末200、201
アプリ実行処理
S3541
アプリへアクセス
S3542
No
クライアント用プログラム受信？
Yes
S3543
UI定義情報取得要求
S3544
No
クライアント用UI定義情報受信？
Yes
S3545
アクション トリガ？
Yes
No
S3546
入力項目に値あり？
Yes
No
S3547
入力項目の値ありアクション要求生成
S3548
アクション要求送信
S3549
アクション結果受信？
Yes
No
S3550
アクション結果反映
S3551
出力項目の値受信？
Yes
No
S3552
出力項目に表示
S3553
終了？
No
Yes
終了

【図３６】


開発者用端末100
3401
UI定義情報（uiDef.json）
・イニシャルUI等の各種アプリ設定
・UI画面毎のUIコンポーネントの情報
…
3402
アクション記述部分
・JavaScript
・Jsonでの関数定義
アップロード
開発環境300
3411
UI定義情報（uiDef.json）
・イニシャルUI等の各種アプリ設定
・UI画面毎のUIコンポーネントの情報
…
3412
アクション記述部分
・JavaScript
・Jsonでの関数定義
生成
実行環境用プログラム
・実行環境の実行エンジンでの実行用プログラム
3414
アクション記述部分（JavaScript）
3413
生成
3615
クライアント用UI定義情報（uiDef2.json）
・イニシャルUI等の各種アプリ設定
・UI画面毎のUIコンポーネントの情報
・コンポーネントのアクションの識別子
・アクション毎の入出力項目定義
"uiItemsIn": ["INPUT_1", "INPUT_2"],
"uiItemsOut": ["RESULT"]
（アクションのJavascriptは持たない）
デプロイ
アプリユーザー用端末200、201
3635
クライアント用UI定義情報（uiDef2.json）
・イニシャルUI等の各種アプリ設定
・UI画面毎のUIコンポーネントの情報
・コンポーネントのアクションの識別子
・アクション毎の入出力項目定義
"uiItemsIn": ["INPUT_1", "INPUT_2"],
"uiItemsOut": ["RESULT"]
（アクションのJavascriptは持たない）
ダウンロード
実行環境400
3421
UI定義情報（uiDef.json）
・イニシャルUI等の各種アプリ設定
・UI画面毎のUIコンポーネントの情報
…
3422
アクション記述部分
・JavaScript
・Jsonでの関数定義
実行環境用プログラム
・実行環境の実行エンジンでの実行用プログラム
3423
3424
アクション記述部分（JavaScript）
3625
クライアント用UI定義情報（uiDef2.json）
・イニシャルUI等の各種アプリ設定
・UI画面毎のUIコンポーネントの情報
・コンポーネントのアクションの識別子
・アクション毎の入出力項目定義
"uiItemsIn": ["INPUT_1", "INPUT_2"],
"uiItemsOut": ["RESULT"]
（アクションのJavascriptは持たない）

フロントページの続き

東京都港区港南２丁目１６番６号 キヤノンＩＴソリューションズ株式会社内

Ｆターム（参考） 5B376 BC07 BC43 BC49
5E555 AA02 AA04 AA09 AA12 AA79 BA02 BA06 BA69 BB02 BB06
BC08 CA24 CB02 CB33 CB34 CB41 CB42 CB44 CB52 CC11 DB04
DB16 DB18 DB20 DC05 DC19 DC21 DC25 DC35 DD11 EA11 FA00