

(19)日本国特許庁(JP)

(12)公開特許公報(A)

(11)公開番号
特開2024-76010
(P2024-76010A)

(43)公開日 令和6年6月5日(2024.6.5)

(51)国際特許分類
G 0 6 F 8/34 (2018.01)

F I
G 0 6 F 8/34

テーマコード (参考)
5 B 3 7 6

審査請求		未請求	請求項の数	15	O L	(全89頁)
(21)出願番号	特願2022-187340(P2022-187340)	(71)出願人	390002761	最終頁に続く		
(22)出願日	令和4年11月24日(2022.11.24)	(71)出願人	キヤノンマーケティングジャパン株式会社			
(特許庁注 : 以下のものは登録商標)		(71)出願人	592135203			
1 . J A V A S C R I P T		(71)出願人	キヤノンITソリューションズ株式会社			
		(74)代理人	100189751			
		(74)代理人	弁理士 木村 友輔			
		(74)代理人	100227857			
		(74)代理人	弁理士 中山 圭			
		(72)発明者	柴本 文洋			
		(72)発明者	東京都区港区港南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			

(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】表への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仕組みを提供する。

【解決手段】 表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。

【選択図】 図 3 1



【特許請求の範囲】**【請求項 1】**

表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システム。

【請求項 2】

前記制御手段は、前記第 2 の操作に応じて、前記表のカラム毎に異なるコンポーネントとしての種別を設定可能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 3】

前記表のカラムに設定可能なコンポーネントとしての種別は、Text Input、Number Input、ComboBox、Multi Select、CheckBox、Data Picker、Link、Button、IconButtonのうち少なくとも 1 つを含む選択肢のいずれかであ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】

前記設定画面においてカラムに対して設定可能な設定項目の少なくとも一部が、当該カラムに設定されたコンポーネントとしての種別に応じて異な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 5】

前記表示制御手段は、前記表に含まれる複数のカラムのそれぞれに対応する選択肢を前記設定画面に表示するように制御し、前記選択肢のうち選択された選択肢に対応するカラムに対する設定を行うための要素画面を前記設定画面に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、前記表の全体にかかる設定を行うための特定の選択肢を前記設定画面に表示するように制御し、前記特定の選択肢が選択されたことに応じて、前記表の全体にかかる設定を行うための要素画面を前記設定画面に表示するように制御することを特徴とする請求項 5 に記載の情報処理システム。

【請求項 7】

前記第 2 の操作は、前記要素画面に対する操作であることを特徴とする請求項 5 に記載の情報処理システム。

【請求項 8】

前記第 1 の操作は、カラムの追加を指示するための、前記設定画面に表示された表示アイテムに対する操作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 9】

前記表示制御手段は、前記表示アイテムに対する操作に応じて、追加するカラムの ID とラベルの設定を受け付けるフォームを表示するように制御することを特徴とする請求項 8 に記載の情報処理システム。

【請求項 10】

前記表は、アプリケーションソフトウェアの開発画面に、構築後の前記アプリケーションソフトウェアの画面に表示すべきコンポーネントとして配置された表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 11】

前記第 2 の操作による設定内容を示す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、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ソフトウェア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 10 に記載の情報

10

20

30

40

50

処理システム。

【請求項 1 2】

前記表示制御手段はさらに、前記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領域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可能であり、複数のカラム毎のアクションの入力領域にそれぞれ対応する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領域を表示し、

前記制御手段は、表示された前記入力領域に対する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 1 3】

前記表示制御手段は、前記選択された表に関する第 1 の選択肢と第 2 の選択肢とを表示し、前記第 1 の選択肢が選択された場合には前記選択された表の前記設定画面を表示するように制御し、前記第 2 の選択肢が選択された場合には前記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領域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ことを特徴とする請求項 1 2 に記載の情報処理システム。

【請求項 1 4】

表の選択を受け付ける受付ステップと、

選択された表の設定画面を表示するように制御する表示制御ステップと、

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 1 5】

少なくとも 1 つのコンピュータを、請求項 1 乃至 1 3 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も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に表（データグリッド）を配置する方法が提案されている。具体的には、以下が開示されている。画面エディタ上にデータグリッドの部品（コンポーネント）をドラッグアンドドロップ操作等により配置する。データグリッドの部品を配置したタイミングでは、単なる格子状のグリッドが表示されるため、この表にどのデータモデルのどの項目を表示させるかを設定する。この項目の設定もデータモデルをドラッグアンドドロップすることにより設定される。

【先行技術文献】

【特許文献】

【0004】

【特許文献 1】特開 2018 - 010616 号公報

【発明の概要】

【発明が解決しようとする課題】

【0005】

10

20

30

40

50

ユーザーは、表に対して、カラム（列）を追加し、追加したカラムの詳細設定を行うといった一連の操作を行う。この一連の操作において、例えば、カラムの追加は表の設定画面から行い、カラムの詳細設定は配置した表のカラムに対応する位置をクリックなどしてカラムを指定してから行うものとする。別々の操作感での操作となり、操作の煩雑化や操作手数の増大を招く。また、特許文献 1 で表にデータモデルの項目を追加する場合のように、表に対して部品をドラッグアンドドロップすることによってカラムの追加や詳細設定を行うことも考えられるが、表に対してどの部品をドラッグアンドドロップで追加できるのかがわかりにくく、ユーザーの混乱を招く可能性もある。

【 0 0 0 6 】

そこで本発明は、上記課題に鑑み、表への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仕組みを提供することを目的とする。 10

【課題を解決するための手段】

【 0 0 0 7 】

本発明の情報処理システムは、
表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 20
を有することを特徴とする。

【発明の効果】

【 0 0 0 8 】

本発明によれば、表への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。

【図面の簡単な説明】

【 0 0 0 9 】

【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 1 0 0、アプリユーザー用端末 2 0 0、アプリユーザー用端末 2 0 1 のハードウェアブロック図である。 30

【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 App Bar 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 1 0】アクションボード処理における表示例である。

【図 1 1】開発環境 3 0 0 が生成するソースコードの例である。

【図 1 2】画面切替処理のフローチャートである。 40

【図 1 3】アプリのユーザー情報の表示にかかる、開発者用端末 1 0 0、開発環境 3 0 0、実行環境 4 0 0 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 1 4】（ a ）開発者情報 3 0 1 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 4 1 1 の具体例である。（ c ）シングルテナント実行環境のユーザー情報の具体例である。

【図 1 5】ユーザー情報の表示例である。

【図 1 6】ワークフロー処理のフローチャートである。

【図 1 7】ワークフロー処理にかかる表示例である。

【図 1 8】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。 50

- 【図 19】キャンパ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】C R U D 生成処理にかかる表示例である。
- 【図 21】C R U D 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 D B 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 U I 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パ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 U I 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 U I 定義情報の遷移図である。
- 【発明を実施するための形態】

10

20

【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せられてもよい。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。

30

【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、P C）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。

40

【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 W e b サービス（クラウドサービス）を組み合わせで構築される。

【0015】

開発者情報 301 は、開発者のアカウント I D となるメールアドレス（ユーザー I D）、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

50

報である。開発者情報 3 0 1 は開発環境 3 0 0 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 3 0 1 の詳細については図 1 4 (a) を用いて後述する。

【 0 0 1 6 】

実行エンジン 3 0 2 は、開発環境 3 0 0 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 3 0 3 とメモリ 3 0 4 を含む。プロセッサ 3 0 3 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 3 0 4 は、プロセッサ 3 0 3 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 3 0 0 が実行するものとして説明するものは、実行エンジン 3 0 2 が実行する。すなわち、メモリ 3 0 4 に記録されたプログラムを、プロセッサ 3 0 3 が開発環境 3 0 0 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。

10

【 0 0 1 7 】

配信エンジン 3 0 5 は、開発環境 3 0 0 にアクセスした開発者用端末 1 0 0 に対して、開発者用端末 1 0 0 において実行すべきクライアント用プログラム 3 2 2 (HTML のソースコードや J a v a S c r i p t 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 3 2 2 は、開発環境 3 0 0 に含まれるストレージ 3 2 0 に予め記録されている。

【 0 0 1 8 】

ストレージ 3 2 0 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 3 2 2 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 3 2 3、開発者 B のための領域である開発者 B 領域 3 2 4、開発者 C のための領域である開発者 C 領域 3 2 5 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、開発者 A 領域 3 2 3 にはアプリ定義 3 2 3 a (図 3 4 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。

20

【 0 0 1 9 】

開発者は、開発者用端末 1 0 0 のブラウザソフトから、開発環境 3 0 0 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 3 0 0 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 3 2 2 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 3 0 0 に送信する。開発環境 3 0 0 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 3 0 0 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 1 0 0 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。

30

40

【 0 0 2 0 】

実行環境 4 0 0 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 4 0 0 はマルチテナント実行環境 4 1 0 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 4 5 0、4 6 0、4 7 0) を含む。実行環境 4 0 0 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 4 0 0 には、開発者用端末 1 0 0 と開発環境 3 0 0 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 2 0 0、2 0 1 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 4 0 0 にアクセスする。そして、アプリユーザー用端末 2 0 0、2 0

50

1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。

10

【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414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。

20

【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。

30

【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。

40

【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

50

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、いくつかの制限があり、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２

10

20

30

40

50

01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。

10

【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。

20

【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ける表示先は、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。

30

【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。

40

50

【 0 0 3 7 】

< 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 1 0 0 から開発環境 3 0 0 にログインして U I エディタを表示するまでの処理である。開発者用端末 1 0 0 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 U R L 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 1 0 0 の C P U 1 0 1 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 1 0 3 に記録されたプログラムと、開発環境 3 0 0 から受信したクライアント用プログラム 3 2 2 とを、メモリ 1 0 2 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 1 0 0 が実行する処理として記載したものは、開発者用端末 1 0 0 の C P U 1 0 1 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 1 0 3 に記録されたプログラムと、開発環境 3 0 0 から受信したクライアント用プログラム 3 2 2 とを、メモリ 1 0 2 に展開して実行する処理であるものとする。

10

【 0 0 3 8 】

開発者用端末 1 0 0 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 U R L を指定してアクセスすると、開発環境 3 0 0 の配信エンジン 3 0 5 がアクセスを検知し、アクセス元の開発者用端末 1 0 0 にクライアント用プログラム 3 2 2 を送信する。

【 0 0 3 9 】

S 3 0 1 では、開発者用端末 1 0 0 は、開発環境 3 0 0 から送信されたクライアント用プログラム 3 2 2 を受信したか否かを判定する。受信していなければ S 3 0 1 で受信を待ち、受信したら S 3 0 2 に進む。

20

【 0 0 4 0 】

S 3 0 2 では、開発者用端末 1 0 0 は、開発環境 3 0 0 から受信したクライアント用プログラム 3 2 2 をメモリ 1 0 2 に記録する。

【 0 0 4 1 】

S 3 0 3 では、開発者用端末 1 0 0 は、クライアント用プログラム 3 2 2 に従い、ログイン画面をディスプレイ 1 0 5 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 I D 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。

30

【 0 0 4 2 】

S 3 0 4 では、開発者用端末 1 0 0 は、ログイン画面の開発者 I D 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 1 0 6 を用いて行われる。開発者 I D 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 I D (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。

【 0 0 4 3 】

S 3 0 5 では、開発者用端末 1 0 0 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 1 0 6 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 S 3 0 6 に進み、そうでない場合には S 3 0 7 に進む。

40

【 0 0 4 4 】

S 3 0 6 では、開発者用端末 1 0 0 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 2 2 (a) を用いて後述する。

【 0 0 4 5 】

50

S 3 0 7では、開発者用端末 1 0 0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S 3 0 8に進み、そうでない場合にはS 3 0 4に戻る。

【 0 0 4 6 】

S 3 0 8では、開発者用端末 1 0 0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 3 0 0に送信する。送信後、開発環境 3 0 0において認証処理が行われるため、その結果を待つ。

【 0 0 4 7 】

S 3 0 9では、開発者用端末 1 0 0は、開発環境 3 0 0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S 3 0 4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S 3 1 0へ進む。

【 0 0 4 8 】

S 3 1 0では、開発者用端末 1 0 0は、開発環境 3 0 0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 3 0 0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 1 0 0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S 3 1 1に進み、そうでない場合にはS 3 0 9に戻る。

【 0 0 4 9 】

S 3 1 1では、開発者用端末 1 0 0は、ログイン後のアプリの開発画面として、S 3 1 0で受信した実行環境リストをメモリ 1 0 2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 1 0 5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、S 3 1 1以降に開発者用端末 1 0 0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。

【 0 0 5 0 】

図 5（a）に、S 3 1 1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 4 1 0に対応する選択肢 5 5 1と、シングルテナント実行環境に対応する選択肢 5 5 2の2つの選択肢が表示されている。開発者ユー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 5 5 3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。

【 0 0 5 1 】

S 3 1 2では、開発者用端末 1 0 0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 5 5 3が押下された場合にはS 3 1 3に進み、そうでない場合にはS 3 1 2で実行環境の選択を待つ。

【 0 0 5 2 】

S 3 1 3では、開発者用端末 1 0 0は、S 3 1 2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 1 0 2に記録するとともに、開発環境 3 0 0に送信する。開発環境 3 0 0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 3 2 0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。

【 0 0 5 3 】

S 3 1 4では、開発者用端末 1 0 0は、開発環境 3 0 0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S 3 1 5に進み、そうでない場合にはS 3 1 4でアプリ情報の受信を待つ。

【 0 0 5 4 】

10

20

30

40

50

S 3 1 5 では、開発者用端末 1 0 0 は、受信したアプリ情報をメモリ 1 0 2 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（アプリー覧）をディスプレイ 1 0 5 に表示する。

【 0 0 5 5 】

S 3 1 6 では、開発者用端末 1 0 0 は、アプリー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 S 3 1 7 に進み、そうでない場合は S 3 2 0 に進む。

【 0 0 5 6 】

S 3 1 7 では、開発者用端末 1 0 0 は、アプリー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 1 0 2 に記録するとともに、開発環境 3 0 0 に送信する。開発環境 3 0 0 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 3 2 0 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 1 0 0 に送信する。

10

【 0 0 5 7 】

S 3 1 8 では、開発者用端末 1 0 0 は、開発環境 3 0 0 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 1 0 2 に記録して S 3 1 9 に進み、そうでない場合には S 3 1 8 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、J s o n 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 J s o n 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 1 0 5 に表示を行う場合には、メモリ 1 0 2 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 1 0 2 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 1 0 2 に記録された最新の定義情報を開発環境 3 0 0 に送信し、ストレージ 3 2 0 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 3 0 0 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。

20

【 0 0 5 8 】

S 3 1 9 では、開発者用端末 1 0 0 は、ディスプレイ 1 0 5 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）のキャンパス（UI 画面の編集領域）を表示する。デスクトップ用であれば 1 6 : 9 の矩形のキャンパスとし、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パスとする。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 S 4 0 1 で行われる）。また、キャンパスには、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パスには何も表示しないようにしても良い。S 3 1 9 の処理でログイン処理を終了し、続いて、図 4 の S 4 0 1 へ進む。

30

40

【 0 0 5 9 】

一方、S 3 2 0 では、開発者用端末 1 0 0 は、アプリー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 S 3 2 1 に進み、そうでない場合には S 3 1 6 に戻る。

【 0 0 6 0 】

S 3 2 1 では、開発者用端末 1 0 0 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

50

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などのアプリケーション用端末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ケーション用端末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。

10

【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。

20

【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。

30

【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。

40

【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

50

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な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。 10

【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。 20

【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。 30

【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。 40

【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 50

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。

10

【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。

20

【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。

30

【0080】

選択中UI画面ボックス503には、キャンバス530で編集中的UI画面を表す情報として、編集中的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。

40

【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。

50

【 0 0 8 2 】

サブメニュー領域 5 2 0 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、U I 画面ボタン 5 1 2 の下位階層メニューとして U I コンポーネント一覧 (U I 部品一覧) が表示されている例である。

【 0 0 8 3 】

キャンバス 5 3 0 は、選択中アプリの選択中の U I 画面 (選択中 U I 画面ボックス 5 0 3 に表示されている U I 画面名の U I 画面) のレイアウト編集領域である。図 5 (b) のキャンバス 5 3 0 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 5 2 0 に表示された U I 部品一覧の中から任意の U I 部品 (U I コンポーネント) を選び、キャンバス領域 5 3 0 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 5 3 0 に配置された U I 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 5 3 0 に配置された U I 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 U I 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 5 3 0 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 U I 画面がロードされた場合に実行 (その U I 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。

【 0 0 8 4 】

図 5 (b) は、アプリ名「 U I 1 」のアプリの、 U I 画面名「 u i 1 」のキャンバス 5 3 0 に、 U I 部品として、パイチャート 5 3 1 (p i e c h a r t 、円グラフ) 、ボタン 5 3 2 、テキストフィールド 5 3 3 、 5 3 4 、アウトプットフィールド 5 3 5 、 (O u t p u t F i e l d) 、タブ部品 5 3 6 を配置した例である。操作パス 5 3 1 a は、選択されている U I 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 5 3 1 が選択されていることを示している。

【 0 0 8 5 】

図 4 に、 U I 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 1 0 0 が実行する処理であり、図 3 の S 3 1 9 または S 3 2 3 の後に続けて行われる処理である。

【 0 0 8 6 】

S 4 0 1 では、開発者用端末 1 0 0 は、選択中アプリの定義情報に基づき、サブメニュー領域 5 2 0 に選択中アプリの U I 画面一覧を選択肢として表示する。この U I 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、アプリユーザー端末 2 0 0 、 2 0 1 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 U I 画面一覧では、新規 U I 画面を追加する操作や、 U I 画面の削除操作も受付可能である。

【 0 0 8 7 】

S 4 0 2 では、開発者用端末 1 0 0 は、 S 4 0 1 で表示された U I 画面一覧のうちいずれかの U I 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 U I 画面が選択された場合は S 4 0 4 に進み、そうでない場合には S 4 0 3 に進む。

【 0 0 8 8 】

S 4 0 3 では、メインメニュー領域 5 1 0 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 5 1 0 に表示された選択肢が選択された場合には U I エディタ処理を終了し、 1 1 2 で後述する画面切替処理へ進む。そうでない場合には S 4 0 2 に戻る。

【 0 0 8 9 】

S 4 0 4 では、開発者用端末 1 0 0 は、メモリ 1 0 2 に記録した選択中アプリの定義情

10

20

30

40

50

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に表示される。S402 で選択された UI 画面が、テンプレー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を判定する。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

10

20

30

40

50

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 S 4 1 2 に進み、そうでない場合には S 4 1 5 に進む。

【 0 0 9 8 】

S 4 1 2 では、開発者用端末 1 0 0 は、S 4 1 1 でクリックされた U I 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 5 3 1 a や操作パス 6 0 1 a である。

【 0 0 9 9 】

S 4 1 3 では、開発者用端末 1 0 0 は、S 4 1 1 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 S 4 3 1 に進み、タブ部分である場合には S 4 1 4 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。

【 0 1 0 0 】

S 4 1 4 では、開発者用端末 1 0 0 は、タブ部品の要素画面領域の表示を、S 4 1 1 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、S 4 1 1 でタブ部品 5 3 6 のうち、I T E M 1、I T E M 2、I T E M 3 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、S 4 1 2 でタブ部品 5 3 6 に対して操作パスを表示するとともに、要素画面領域 4 3 6 a 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 6 2 0 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 U I 部品 6 1 1、6 1 2 が非表示となり、代わりに、クリックされたタブ 6 2 0 に対応する要素画面に配置されている U I 部品 6 2 1 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 U I 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 U I 部品を配置したい場合、配置したい U I 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。

【 0 1 0 1 】

なお、本実施形態では説明の簡略化のため、S 4 0 9、S 4 1 0、S 4 1 3、S 4 1 4 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、U I 部品内の一部の表示内容が当該 U I 部品への操作に応じて切り替わる所定種別の U I 部品であれば、タブ部品と同様に S 4 0 9、S 4 1 0、S 4 1 3、S 4 1 4 の処理を適用可能である。

【 0 1 0 2 】

一方、タブ部品などの所定種別の U I 部品ではない他の U I 部品の場合には、キャンバス 5 3 0 上で U I 部品が選択されたことに応じて当該 U I 部品内の表示内容を切り替えることはしない。U I 部品の選択に応じて行うのは操作パ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 5 3 0 でボタン 5 3 2 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 5 3 2 のアクション (構築されたアプリでボタン 5 3 2 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 5 3 0 でテキストフィールド 5 3 3 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 5 3 3 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。

【 0 1 0 3 】

図 6 (c)、図 6 (d)、図 6 (e) を用いて、タブ部品と同様に S 4 0 9、S 4 1 0、S 4 1 3、S 4 1 4 の処理を適用可能な種別の U I 部品の例として A p p B a r の例を説明する。図 6 (c) は、キャンバス 5 3 0 に、それぞれ U I 部品である A p p B a r 6 5 0、T e x t F i e l d 6 6 1、B u t t o n 6 6 2 を配置した場合の例である。A p p B a r 6 5 0 は 1 つの U I 部品であり、A p p B a r 6 5 0 の中に要素アイコン 6 5 1 と要素アイコン 6 5 2 が含まれる。要素アイコン 6 5 1 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 6 5 2 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 5 3 0 において

、配置済みの App Bar 650 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 App Bar 650 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)、App Bar 650 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、App Bar 650 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。

10

【0105】

配置済みの App Bar 650 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)、App Bar 650 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、App Bar 650 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。ポップアップメニュー 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、ポップアップメニュー 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。

20

【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。

30

【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。

40

【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

50

なボタン 5 3 2 (指定 U I 部品) に関するコンテキストメニュー 5 4 0 を表示する。コンテキストメニュー 5 4 0 には、選択肢となるメニュー項目としてプロパティ 5 4 1、アクション 5 4 2、消去 5 4 3 が表示される。プロパティ 5 4 1 が選択されると、ボタン 5 3 2 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 5 3 2 に表示するボタン名 (ラベル)、ボタン 5 3 2 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 5 4 2 が選択されると、ボタン 5 3 2 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 J a v a S c r i p t 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 5 3 2 (指定 U I 部品) が押下された場合に実行すべき処理である。消去 5 4 3 が選択されると、キャンバス 5 3 0 からボタン 5 3 2 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。

10

【 0 1 0 9 】

S 4 2 1 では、開発者用端末 1 0 0 は、保存ボタン 5 0 4 が押下されたか否かを判定する。保存ボタン 5 0 4 が押下された場合には S 4 2 2 に進み、そうでない場合には S 4 2 3 へ進む。S 4 2 2 では、開発者用端末 1 0 0 は、メモリ 1 0 2 に記録している編集中的アプリケーションの定義情報を開発環境 3 0 0 に送信する。開発環境 3 0 0 は、定義情報を受信すると図 3 3 (a) または図 3 5 (a) で後述する保存処理を行う。

【 0 1 1 0 】

S 4 2 3 では、開発者用端末 1 0 0 は、プレビューボタン 5 0 5 が押下されたか否かを判定する。プレビューボタン 5 0 5 が押下された場合と判定した場合には S 4 2 4 へ進み、そうでない場合には S 4 2 5 へ進む。S 4 2 4 では、開発者用端末 1 0 0 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 5 0 3 を非表示とし、メモリ 1 0 2 に記録している定義情報または開発環境 3 0 0 に記録されている定義情報に基づいて、キャンバス 5 0 3 で編集している U I 画面について、構築されたアプリケーションでその U I 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、U I 部品を操作するための操作パスは表示されない。また、U I エディタ画面では実行されない (キャンバス 5 3 0 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 U I 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、アクションボード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ず、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。

20

30

【 0 1 1 1 】

S 4 2 5 では、開発者用端末 1 0 0 は、デプロイボタン 5 0 6 が押下されたか否かを判定する。デプロイボタン 5 0 6 が押下された場合には S 4 2 6 へ進み、そうでない場合には S 4 2 7 へ進む。S 4 2 6 では、開発者用端末 1 0 0 は、デプロイ処理を実行する。デプロイ処理については図 2 4 (a) を用いて後述する。なおデプロイボタン 5 0 6 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 5 0 1 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 S 3 1 1)、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、作業を効率化することができる。また、開発環境 3 0 0 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 1 0 0 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。

40

【 0 1 1 2 】

50

S 4 2 7では、開発者用端末 1 0 0 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 5 0 1 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 S 4 2 8 へ進み、そうでない場合には S 4 2 9 へ進む。S 4 2 8では、開発者用端末 1 0 0 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 1 0 2 に記録するとともに、開発環境 3 0 0 に送信する。また、選択中実行環境ボックス 5 0 1 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 3 0 0 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 3 2 0 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。

10

【 0 1 1 3 】

S 4 2 9では、開発者用端末 1 0 0 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 5 0 2 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 S 3 1 7 に進み、変更後の選択中アプリに基づいて S 3 1 7 以降の処理を実行する。すなわち、サブメニュー領域 5 2 0、キャンバス 5 3 0 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 S 4 3 0 へ進む。

【 0 1 1 4 】

S 4 3 0では、開発者用端末 1 0 0 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 5 0 3 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 S 4 0 4 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 5 3 0 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 S 4 3 1 へ進む。

20

【 0 1 1 5 】

S 4 3 1では、開発者用端末 1 0 0 は、メインメニュー領域 5 1 0 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 5 1 0 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 1 2 で後述する画面切替処理へ進む。そうでない場合には S 4 0 6 に戻る。

【 0 1 1 6 】

30

< 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 S 4 2 0 の詳細フローチャートである。

【 0 1 1 7 】

S 7 0 1では、開発者用端末 1 0 0 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、S 4 1 9 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 S 7 1 0 に進み、そうでない場合には S 7 0 2 に進む。

【 0 1 1 8 】

40

S 7 0 2では、開発者用端末 1 0 0 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、S 4 1 9 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 5 3 0 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 S 7 0 3 に進み、そうでない場合には S 7 0 4 に進む。

【 0 1 1 9 】

S 7 0 3では、開発者用端末 1 0 0 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ストメニュー処理については図 1 9 を用いて後述する。

【 0 1 2 0 】

50

S 7 0 4では、開発者用端末 1 0 0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、S 4 1 9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S 7 0 5に進み、そうでない場合にはS 7 0 6に進む。

【0 1 2 1】

S 7 0 5では、開発者用端末 1 0 0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ストメニュー処理については図 3 0を用いて後述する。

【0 1 2 2】

S 7 0 6では、開発者用端末 1 0 0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。 10

【0 1 2 3】

S 7 1 0では、開発者用端末 1 0 0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S 7 2 1に進み、そうでない場合にはS 7 1 1に進む。

【0 1 2 4】

S 7 1 1では、開発者用端末 1 0 0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。 20

【0 1 2 5】

S 7 1 2では、開発者用端末 1 0 0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 5 4 1）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S 7 1 3に進み、そうでない場合にはS 7 1 4へ進む。

【0 1 2 6】

S 7 1 3では、開発者用端末 1 0 0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 5 3 0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 5 3 0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。 30

【0 1 2 7】

S 7 1 4では、開発者用端末 1 0 0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 5 4 2）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S 7 1 5に進み、そうでない場合にはS 7 1 6へ進む。

【0 1 2 8】

S 7 1 5では、開発者用端末 1 0 0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。

【0 1 2 9】

S 7 1 6では、開発者用端末 1 0 0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 5 4 3）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S 7 1 7に進み、そうでない場合にはS 7 1 8へ進む。 40

【0 1 3 0】

S 7 1 7では、開発者用端末 1 0 0は、指定UI部品を消去し、メモリ 1 0 2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 5 3 0において指定UI部品は消去（削除）されて非表示となる。

【0 1 3 1】

S 7 1 8では、開発者用端末 1 0 0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S 7 1 9 50

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 S 7 2 0 へ進む。

【 0 1 3 2 】

S 7 2 0 では、開発者用端末 1 0 0 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 S 7 1 2 に戻る。

【 0 1 3 3 】

一方、S 7 2 1 では、開発者用端末 1 0 0 は、指定 UI 部品に関するコンテキストメニューとして、図 2 6 (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 2 6 (d) の雛形部品 2 6 3 4 ）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 5 4 0 ）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。

10

【 0 1 3 4 】

S 7 2 2 では、開発者用端末 1 0 0 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 S 7 2 3 に進み、そうでない場合には S 7 2 4 へ進む。S 7 2 3 の処理は S 7 1 3 と同様である。

20

【 0 1 3 5 】

S 7 2 4 では、開発者用端末 1 0 0 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 S 7 2 2 に戻る。

【 0 1 3 6 】

< 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ドを表示する。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 1 0 5 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 5 2 0 に UI 部品一覧が表示され、キャンバス 9 0 0 に編集対象の UI 画面が表示されている。キャンバス 9 0 0 には、ボタンである UI 部品 9 0 1 と、その他の UI 部品が配置されている。UI 部品 9 0 1 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。

30

【 0 1 3 7 】

図 9 (b) は、UI 部品 9 0 1 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 9 1 0 は、キャンバス 9 0 0 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、アクションボード 9 1 0 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 9 1 0 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 9 1 0 に、操作部 1 0 6 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 J a v a S c r i p t 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（J a v a S c r i p t 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、

40

50

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。

10

【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。

20

【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。

30

【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。

40

【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】

50

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。

10

【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。

20

30

40

【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、どの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

50

）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 9 2 0 に表示される関数は、アクションボード 9 1 0 の対象となっている指定 UI 部品（図示の例では UI 部品 9 0 1 ）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 9 1 0 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 9 1 0 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 9 1 0 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 9 1 1 や関数一覧 9 2 0 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。

【 0 1 4 6 】

図 1 0 (d) は、アクションボード 9 1 0 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 9 6 0 で示した部分に、創作関数である「 r e s t 0 1 」（関数一覧 9 2 0 の関数 9 2 3 ）を用いている。図 1 0 (d) のアクションボード 9 1 0 に図示したコードの文字列では、創作関数である r e s t 0 1 の詳細は定義されていない。また、開発者が r e s t 0 1 の設定を R E S T 関数設定画面 9 4 0 で行った際にも、プログラミング言語での r e s t 0 1 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 3 0 0 にアップロードして保存した場合、開発環境 3 0 0 が、後述する図 3 3 (a) または図 3 5 (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 9 1 0 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 3 0 0 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。

【 0 1 4 7 】

図 1 1 に、図 1 0 (d) のアクションボード 9 1 0 に記載された文字列と、R E S T 関数設定画面 9 4 0 で設定された r e s t 0 1 の設定内容に基づいて、開発環境 3 0 0 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 ~ 7 5 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 1 1 の例は、プログラミング言語での r e s t 0 1 の詳細の定義を含む 7 5 行のソースコードである。しかし開発者自身は 7 5 行もの入力をする必要はない。R E S T 関数設定画面 9 4 0 での設定を行い、図 1 0 (d) のアクションボード 9 1 0 に図示した分量（7 行）の文字列を記述すれば、図 1 1 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。

【 0 1 4 8 】

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 S 7 1 5

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。

【 0 1 4 9 】

S 8 0 1 では、開発者用端末 1 0 0 は、ディスプレイ 1 0 5 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ド 9 1 0 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 9 1 0 を含む画面を表示する。

【 0 1 5 0 】

S 8 0 2 では、開発者用端末 1 0 0 は、アクションボード 9 1 0 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 9 1 0 を選択した状態で操作部 1 0 6 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 S 8 0 3 に進み、そうでない場合には S 8 1 0 に進む。

【 0 1 5 1 】

S 8 0 3 では、開発者用端末 1 0 0 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 9 1 0 に、入力された文字を表示する。

【 0 1 5 2 】

S 8 0 4 では、開発者用端末 1 0 0 は、S 8 0 3 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 S 8 0 5 に進み、そうでない場合には S 8 0 8 に進む。

【 0 1 5 3 】

S 8 0 5 では、開発者用端末 1 0 0 は、S 8 0 4 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 9 1 1 が表示される。なお、コードアシスト欄 9 1 1 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。

【 0 1 5 4 】

S 8 0 6 では、開発者用端末 1 0 0 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 S 8 0 7 に進み、選択された選択肢の関数名をコードアシスト欄 9 1 1 に表示する。例えば、図 10 (b) の状態からコードアシスト欄 9 1 1 の選択肢が選択されたことに応じて、アクションボード 9 1 0 の 5 行目に示すように、「s q l 0 1」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 S 8 0 8 に進む。

【 0 1 5 5 】

S 8 0 8 では、開発者用端末 1 0 0 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 S 8 0 3 に進み、そうでない場合には S 8 0 2 に進む。

【 0 1 5 6 】

S 8 1 0 では、開発者用端末 1 0 0 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 9 2 1 が押下されたか否かを判定する。関数追加ボタン 9 2 1 が押下された場合には S 8 1 1 に進み、そうでない場合には S 8 2 0 に進む。

【 0 1 5 7 】

S 8 1 1 では、開発者用端末 1 0 0 は、創作関数を追加するための追加ダイアログ 9 3 0 を表示し、追加ダイアログ 9 3 0 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、S A V E ボタン 9 3 3 が押下されると S 8 1 2 に進む。

10

20

30

40

50

【 0 1 5 8 】

S 8 1 2 では、開発者用端末 1 0 0 は、サブメニュー領域の関数一覧 9 2 0 に、追加ダイアログの関数名入力欄 9 3 2 に入力された関数名を追加で、未完マーク 9 2 9 を付した状態で表示する。

【 0 1 5 9 】

S 8 1 3 では、開発者用端末 1 0 0 は、追加ダイアログの種別選択欄 9 3 1 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 S 8 1 4 に進み、そうでない場合には S 8 1 5 に進む。S 8 1 4 では、開発者用端末 1 0 0 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。

【 0 1 6 0 】

S 8 1 5 では、開発者用端末 1 0 0 は、追加ダイアログの種別選択欄 9 3 1 で選択された種別が S Q L であるか否かを判定する。S Q L である場合には S 8 1 6 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 9 3 1 で選択された種別が R E S T である場合）には S 8 1 7 に進む。S 8 1 6 では、開発者用端末 1 0 0 は、S Q L 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。S Q L 関数設定画面で受け付ける設定内容は図 1 0 (a) を用いて前述したとおりである。

【 0 1 6 1 】

S 8 1 7 では、開発者用端末 1 0 0 は、R E S T 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。R E S T 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。

【 0 1 6 2 】

S 8 1 8 では、開発者用端末 1 0 0 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 S 8 1 9 に進み、そうでない場合には S 8 0 2 に進む。

【 0 1 6 3 】

S 8 1 9 では、開発者用端末 1 0 0 は、設定中の創作関数に関して関数一覧 9 2 0 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 9 1 0 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。

【 0 1 6 4 】

S 8 2 0 では、開発者用端末 1 0 0 は、関数一覧 9 2 0 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 9 2 0 で創作関数が選択された場合には、S 8 1 3 に進み、S 8 1 4、S 8 1 6、S 8 1 7 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）。

【 0 1 6 5 】

S 8 2 1 では、開発者用端末 1 0 0 は、保存の指示操作（例えば保存ボタン 9 1 5 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 S 8 2 2 に進み、そうでない場合には S 8 2 3 へ進む。

【 0 1 6 6 】

S 8 2 2 では、開発者用端末 1 0 0 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 1 0 2 に保持した定義情報に記録するとともに、開発環境 3 0 0 に送信する。

【 0 1 6 7 】

S 8 2 3 では、開発者用端末 1 0 0 は、関数一覧 9 2 0 に表示された、アクションボード 9 1 0 の表示を指示する選択肢 9 2 2 が押下されたか否かを判定する。選択肢 9 2 2 が押下された場合は S 8 0 1 に進み、指定 U I 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 9 2 2 が押下されていない場合には S 8 2 4 へ進む。

10

20

30

40

50

【 0 1 6 8 】

S 8 2 4 では、開発者用端末 1 0 0 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 S 8 0 2 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 画面のキャンパスの表示に切り替え、UI エディタ処理へ戻る。

【 0 1 6 9 】

< 画面切替処理 >

図 1 2 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 5 1 0 に表示された選択肢の選択に応じた処理である。

10

【 0 1 7 0 】

S 1 2 0 1 では、開発者用端末 1 0 0 は、アプリー覧ボタン 5 1 1 が押下されたか否かを判定する。アプリー覧ボタン 5 1 1 が押下された場合は図 3 の S 3 1 5 に進み、そうでない場合は S 1 2 0 2 に進む。

【 0 1 7 1 】

S 1 2 0 2 では、開発者用端末 1 0 0 は、UI 画面ボタン 5 1 2 が押下されたか否かを判定する。UI 画面ボタン 5 1 2 が押下された場合は S 1 2 0 3 に進み、そうでない場合は S 1 2 0 4 へ進む。S 1 2 0 3 では、図 4 の UI エディタ処理を行う。

【 0 1 7 2 】

S 1 2 0 4 では、開発者用端末 1 0 0 は、ワークフローボタン 5 1 3 が押下されたか否かを判定する。ワークフローボタン 5 1 3 が押下されたと判定した場合は S 1 2 0 5 に進み、そうでない場合は S 1 2 0 6 に進む。

20

【 0 1 7 3 】

S 1 2 0 5 では、開発者用端末 1 0 0 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 1 6 を使って後述する。

【 0 1 7 4 】

S 1 2 0 6 では、開発者用端末 1 0 0 は、設定ボタン 5 1 4 が押下されたか否かを判定する。設定ボタン 5 1 4 が押下された場合には S 1 2 0 7 に進み、そうでない場合は S 1 2 0 9 へ進む。

【 0 1 7 5 】

S 1 2 0 7 では、開発者用端末 1 0 0 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。S 1 2 0 8 では、開発者用端末 1 0 0 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 1 0 2 に記録した定義情報を更新する。S 1 2 0 8 で受け付け可能な設定には、例えば、表示言語の設定や、PWA (P r o g r e s s i v e W e b A p p s) 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。

30

40

【 0 1 7 6 】

S 1 2 0 9 では、開発者用端末 1 0 0 は、環境一覧ボタン 5 1 5 が押下されたか否かを判定する。環境一覧ボタン 5 1 5 が押下された場合には図 3 の S 3 1 1 に進み、そうでない場合には S 1 2 1 0 へ進む。

【 0 1 7 7 】

S 1 2 1 0 では、開発者用端末 1 0 0 は、データベースボタン 5 1 6 が押下されたか否かを判定する。データベースボタン 5 1 6 が押下された場合には S 1 2 1 1 に進み、そうでない場合には S 1 2 1 5 へ進む。データベースボタン 5 1 6 の押下はデータベースへの接続要求である。

【 0 1 7 8 】

50

S 1 2 1 1 では、開発者用端末 1 0 0 は、選択中実行環境がマルチテナント実行環境 4 1 0 であるか否かを判定する。マルチテナント実行環境 4 1 0 である場合には S 1 2 1 3 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 S 1 2 1 2 に進む。

【 0 1 7 9 】

S 1 2 1 2 では、開発者用端末 1 0 0 は、開発環境 3 0 0 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 3 0 0 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 3 0 0 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 4 5 7、4 6 7、4 7 7 などのうちいずれか）に 10
アクセスし、データベースに記録されている情報を取得する。そして、開発環境 3 0 0 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 1 0 0 に送信する。開発者用端末 1 0 0 から、開発環境 3 0 0 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。

【 0 1 8 0 】

S 1 2 1 3 では、開発者用端末 1 0 0 は、ログイン開発者の開発者情報（開発者情報 3 0 1 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 3 0 0 が選択中実行環境であるマルチテナント実行環境 4 1 0 に含まれる DB セット 4 3 0 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情報を取得す 20
る。そして、開発環境 3 0 0 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 1 0 0 に送信する。開発者用端末 1 0 0 から、開発環境 3 0 0 を介さずにマルチテナント実行環境 4 1 0 にアクセスすることは無い。

【 0 1 8 1 】

S 1 2 1 4 では、開発者用端末 1 0 0 は、データベースの管理画面を表示し、S 1 2 1 2 または S 1 2 1 3 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。

【 0 1 8 2 】

S 1 2 1 5 では、開発者用端末 1 0 0 は、ファイルマネージャーボタン 5 1 7 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 5 1 7 が押下された場合には S 1 2 1 6 に進み、そうでない場合には S 1 2 1 8 に進む。 30

【 0 1 8 3 】

S 1 2 1 6 では、開発者用端末 1 0 0 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、S 1 2 1 7 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 1 0 0 のローカル保存領域（記録媒体 1 0 8 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 1 0 0 は、選択されたファイルを 40
開発環境 3 0 0 に送信する。開発環境 3 0 0 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 3 0 0 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 1 0 0 から、開発環境 3 0 0 を介して各種実行環境にアクセスする。開発者用端末 1 0 0 から、開発環境 3 0 0 を介さずにマルチテナント実行環境 4 1 0 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通り、開発者は、開発環境 3 0 0 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 3 0 0 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 3 0 0 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。

【 0 1 8 4 】

また、開発者ユーザーは開発環境 3 0 0 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。

【 0 1 8 5 】

S 1 2 1 8 では、開発者用端末 1 0 0 は、ユーザー管理ボタン 5 1 8 が押下されたか否かを判定する。ユーザー管理ボタン 5 1 8 が押下されたと判定した場合は S 1 2 1 9 に進み、そうでない場合は S 1 2 2 0 に進む。

【 0 1 8 6 】

S 1 2 1 9 では、開発者用端末 1 0 0 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 4 1 1、4 5 1、4 6 1、4 7 1 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 1 4 を用いて後述する。

【 0 1 8 7 】

S 1 2 2 0 では、開発者用端末 1 0 0 は、スナップショットボタン 5 1 9 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 5 1 9 が押下された場合には S 1 2 2 1 に進んでスナップ処理を行い、そうでない場合には S 1 2 2 2 に進む。

【 0 1 8 8 】

S 1 2 2 2 では、開発者用端末 1 0 0 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 S 1 2 0 1 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。

【 0 1 8 9 】

< ユーザー情報表示処理 >

図 1 4 (a) に、開発環境 3 0 0 に保持している開発者情報 3 0 1 の具体例を示す。開発者情報 3 0 1 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、V e r i f y、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。V e r i f y 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 1 4 (a) に示す情報以外であっても良い。

【 0 1 9 0 】

図 1 4 (b) に、マルチテナント実行環境 4 1 0 に保持しているユーザー情報 4 1 1 の具体例を示す。ユーザー情報 4 1 1 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 とするメールアドレス（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 1 4 (b) に示す情報以外であっても良い。例えば、生成日時、更新日時などが記録されていてもよい。

【 0 1 9 1 】

図 1 4 (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 4 5 1、4 6 1、4 7 1 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 1 4 (c 1) に示すユーザー別情報と、図 1 4 (c 2) に示すグループ情報

10

20

30

40

50

の 2 種類が記録される。

【 0 1 9 2 】

図 1 4 (c 1) 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 1 4 (c 1) に示す情報以外であっても良い。例えば、生成日時、更新日時などが記録されていてもよい。

10

【 0 1 9 3 】

図 1 4 (c 2) 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。

【 0 1 9 4 】

図 1 3 (a) に、開発者用端末 1 0 0 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 1 2 の S 1 2 1 8 の詳細フローチャートである。

【 0 1 9 5 】

S 1 3 0 では、開発者用端末 1 0 0 は、開発環境 3 0 0 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。

20

【 0 1 9 6 】

S 1 3 0 2 では、開発者用端末 1 0 0 は、S 1 3 0 1 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 3 0 0 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 S 1 3 0 2 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 S 1 3 0 3 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 1 4 (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 1 4 (c 1)、図 1 4 (c 2) に示したような情報である。

【 0 1 9 7 】

S 1 3 0 3 では、開発者用端末 1 0 0 は、S 1 3 0 2 で受信したユーザー情報に基づいて、ディスプレイ 1 0 5 にユーザー情報を表示する。図 1 5 に、S 1 3 0 3 で表示される、開発者用端末 1 0 0 におけるユーザー情報の表示例を示す。図 1 5 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 1 5 0 1 は、図 1 4 (c 1) 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 1 5 0 2 は、図 1 4 (c 2) 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 1 5 0 1、1 5 0 2 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 1 5 0 1、1 5 0 2 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 1 5 0 5 は、S 1 3 0 2 で受信したユーザー情報 (図示の例では、図 1 4 (c 1) 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 1 5 0 5 において、U s e r n a m e、E m a i l V e r i f i e d の列はそれぞれ、図 1 4 (c 1) 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。D e l e t e 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 1 5 0 3 は、グループ情報 1 5 0 5 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 1 5 0 4 は図 1 5 の画面で編集された内容のグループ情報 1 5 0 5 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。

30

40

【 0 1 9 8 】

S 1 3 0 4 では、開発者用端末 1 0 0 は、S 1 3 0 3 で表示したユーザー情報の編集操

50

作を受け付ける。例えば、ユーザー追加ボタン 1 5 0 3 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アカウントの削除、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。

【 0 1 9 9 】

S 1 3 0 5 では、開発者用端末 1 0 0 は、保存ボタン 1 5 0 4 が押下されたか否かを判定する。保存ボタン 1 5 0 4 が押下された場合は S 1 3 0 6 に進み、そうでない場合には S 1 3 0 7 へ進む。

【 0 2 0 0 】

S 1 3 0 6 では、開発者用端末 1 0 0 は、開発環境 3 0 0 に対して、S 1 3 0 4 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。

10

【 0 2 0 1 】

S 1 3 0 7 では、開発者用端末 1 0 0 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 5 0 1 に対する操作があった場合）には S 1 3 0 8 に進み、そうでない場合には S 1 3 0 9 に進む。

【 0 2 0 2 】

S 1 3 0 8 では、開発者用端末 1 0 0 は、変更後の選択中実行環境をメモリ 1 0 2 に記録するとともに、開発環境 3 0 0 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、S 1 3 0 1 に戻って再度処理を行う。

【 0 2 0 3 】

S 1 3 0 9 では、開発者用端末 1 0 0 は、メインメニュー領域 5 1 0 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 S 1 3 0 4 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。

20

【 0 2 0 4 】

図 1 3 (b) に、図 1 3 (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 3 0 0 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。

【 0 2 0 5 】

S 1 3 1 1 では、開発環境 3 0 0 は、S 1 3 0 1 で開発者用端末 1 0 0 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 S 1 3 1 2 に進み、そうでない場合には S 1 3 1 6 に進む。

30

【 0 2 0 6 】

S 1 3 1 2 では、開発環境 3 0 0 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 4 1 0 であればアクセス先情報 4 2 1 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 4 5 6 b、4 6 6 b、4 7 6 b ... 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 3 0 0 は、選択中実行環境として、開発者情報 3 0 1 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 3 0 4 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、パケット、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 . j s o n」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 3 0 4 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

40

50

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な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。

10

【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。

20

【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S1318に進む。

30

【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。

40

【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

50

行環境 4 0 0 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 4 1 0 であれば動作主は実行エンジン 4 1 2 であり、実行エンジン 4 1 2 に含まれるプロセッサ 4 1 3 が、メモリ 4 1 4 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 4 5 0 であれば動作主は実行エンジン 4 5 2 であり、実行エンジン 4 5 2 に含まれるプロセッサ 4 5 3 が、メモリ 4 5 4 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。

【 0 2 1 5 】

S 1 3 2 1 では、実行環境 4 0 0 は、開発環境 3 0 0 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 S 1 3 1 3 で開発環境 3 0 0 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 S 1 3 2 2 に進み、そうでない場合には S 1 3 2 7 に進む。

10

【 0 2 1 6 】

S 1 3 2 2 では、実行環境 4 0 0 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 1 3 (d) を用いて後述する。

【 0 2 1 7 】

S 1 3 2 3 では、実行環境 4 0 0 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、S 1 3 2 1 に進む。エラーでない場合には S 1 3 2 4 へ進む。

20

【 0 2 1 8 】

S 1 3 2 4 では、実行環境 4 0 0 は、自身がマルチテナント実行環境 4 1 0 であるか否かを判定する。マルチテナント実行環境 4 1 0 である場合には S 1 3 2 6 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 S 1 3 2 5 へ進む。

【 0 2 1 9 】

S 1 3 2 5 では、実行環境 4 0 0 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 3 0 0 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 S 1 3 2 6 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 4 5 0 であれば、ユーザー情報 4 5 1 を開発環境 3 0 0 に送信する。ここで送信したユーザー情報が、開発環境 3 0 0 に S 1 3 1 4 で受信され、S 1 3 1 5 で開発者用端末 1 0 0 に送信される。

30

【 0 2 2 0 】

S 1 3 2 6 では、実行環境 4 0 0 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 3 0 0 に送信する。暗号化された開発者情報が開発環境 3 0 0 から送信され、後述する S 1 3 4 1 で受信され、S 1 3 4 3 でデコード（暗号化解除）され、メモリ 4 1 4 に記憶されている。これが受信した開発者情報である。図 1 4 (b) に示したマルチテナント実行環境 4 1 0 のユーザー情報 4 1 1 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 3 0 0 に送信する。マルチテナント実行環境 4 1 0 のユーザー情報 4 1 1 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 3 0 0 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 3 0 0 に S 1 3 1 4 で受信され、S 1 3 1 5 で開発者用端末 1 0 0 に送信される。

40

50

【 0 2 2 1 】

S 1 3 2 7では、実行環境 4 0 0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 S 1 3 1 7で開発環境 3 0 0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 S 1 3 2 8に進み、そうでない場合には S 1 3 2 1に進む。

【 0 2 2 2 】

S 1 3 2 8では、実行環境 4 0 0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 1 3（d）を用いて後述する。

10

【 0 2 2 3 】

S 1 3 2 9では、実行環境 4 0 0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、S 1 3 2 1に進む。エラーでない場合には S 1 3 3 0へ進む。

【 0 2 2 4 】

S 1 3 3 0では、実行環境 4 0 0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。

20

【 0 2 2 5 】

図 1 3（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 1 3（c）の S 1 3 2 2、S 1 3 2 8の詳細フローチャートである。

【 0 2 2 6 】

S 1 3 4 1では、実行環境 4 0 0は、開発環境 3 0 0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 S 1 3 1 3または S 1 3 1 7で開発環境 3 0 0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 S 1 3 4 3に進み、そうでない場合には S 1 3 4 2に進む。

【 0 2 2 7 】

S 1 3 4 2では、実行環境 4 0 0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 S 1 3 2 3または S 1 3 2 9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。

30

【 0 2 2 8 】

S 1 3 4 3では、実行環境 4 0 0は、S 1 3 4 1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。

【 0 2 2 9 】

S 1 3 4 4では、実行環境 4 0 0は、S 1 3 4 2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 S 1 3 4 2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。

40

【 0 2 3 0 】

なお、実行環境 4 0 0は、開発環境 3 0 0以外からのアクセスは受け付けず、開発環境

50

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こ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。

10

【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。

20

【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。

30

【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。

40

【0236】

50

S 1 6 0 2 では、開発者用端末 1 0 0 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 1 7 1 0 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 1 7 0 0 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 S 1 6 0 3 に進み、そうでない場合には S 1 6 0 4 に進む。

【 0 2 3 7 】

S 1 6 0 3 では、開発者用端末 1 0 0 は、ノードを配置する操作に応じて WF 配置領域 1 7 0 0 に選択されたノードを配置する。

【 0 2 3 8 】

S 1 6 0 4 では、開発者用端末 1 0 0 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ursor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 S 1 6 0 5 に進み、そうでない場合には S 1 6 0 6 に進む。

【 0 2 3 9 】

S 1 6 0 5 では、開発者用端末 1 0 0 は、矢印を描画する操作に応じて WF 配置領域 1 7 0 0 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。

【 0 2 4 0 】

図 1 7 (b) に、WF 配置領域 1 7 0 0 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 1 7 0 0 には、ノード 1 7 0 1 ~ 1 7 0 5 が配置されている。ノード 1 7 0 1 は始点のノードである。ノード 1 7 0 2、1 7 0 3、1 7 0 4 は中間点（ステータス）のノードである。ノード 1 7 0 5 は終点のノードである。矢印 1 7 0 6 は、ノード 1 7 0 1 からノード 1 7 0 2 までの矢印である。矢印 1 7 0 7 は、ノード 1 7 0 2 からノード 1 7 0 3 までの矢印である。矢印 1 7 0 8 は、ノード 1 7 0 3 からノード 1 7 0 4 までの矢印である。矢印 1 7 0 9 は、ノード 1 7 0 4 からノード 1 7 0 5 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 1 7 0 1 ~ 1 7 0 5 がそれぞれ、1 ~ 5 番目の手順となることが示されている。

【 0 2 4 1 】

S 1 6 0 6 では、開発者用端末 1 0 0 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 1 7 0 0 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 S 1 6 0 7 に進み、そうでない場合には S 1 6 0 8 に進む。

【 0 2 4 2 】

S 1 6 0 7 では、開発者用端末 1 0 0 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 1 7 (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。

【 0 2 4 3 】

S 1 6 0 8 では、開発者用端末 1 0 0 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 1 7 0 0 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 S 1 6 0 9 に進み、そうでない場合には S 1 6 1 0 に進む。

【 0 2 4 4 】

S 1 6 0 9 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 1 7 (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 I D (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 (U p d a t e)、申請 (C r e a t)、差戻 (R e m a n d) 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 S e l e c t U s e r と C u r r e n t U s e r 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。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 U I 部品のアクションが自動設定される。

10

【 0 2 4 5 】

S 1 6 1 0 では、開発者用端末 1 0 0 は、ワークフロー生成ボタン 1 7 3 0 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 1 7 3 0 が押下された場合は S 1 6 1 1 に進み、そうでない場合は S 1 6 1 5 に進む。

【 0 2 4 6 】

S 1 6 1 1 では、開発者用端末 1 0 0 は、図 1 7 (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。

20

【 0 2 4 7 】

S 1 6 1 2 では、開発者用端末 1 0 0 は、図 1 7 (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 U I 画面の I D の選択が可能である。U I 画面の I D の選択は、U I 画面の I D 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。

【 0 2 4 8 】

S 1 6 1 3 では、開発者用端末 1 0 0 は、図 1 7 (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。

30

【 0 2 4 9 】

S 1 6 1 4 では、開発者用端末 1 0 0 は、W F 配置領域 1 7 0 0 に配置済みのノード、矢印と、それぞれのプロパティ、及び、S 1 6 1 1 ~ S 1 6 1 4 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 1 0 2 の定義情報に記録する。これによって、U I 画面にワークフローに基づく U I 部品 (U I 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 U I 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 U I 部品が生成される。また、図 1 7 (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 U I 画面のキャンパスにもアクションが設定 (定義) される。キャンパ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 U I 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。

40

【 0 2 5 0 】

S 1 6 1 5 では、開発者用端末 1 0 0 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 5 1 0 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 5 1 0 に含まれるいずれかの選択肢が選択された場合には図 1 6 の処理を終了し、図 1 2 で前述した画面切替処理へ進む。そうでない場合は S 1 6 0 2 に進む。

50

【 0 2 5 1 】

図 1 8 (a) に、図 1 6 の S 1 6 1 4 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 5 1 6 を押下して表示させた場合の表示例を示す。すなわち、図 1 8 (a) は、図 1 2 の S 1 2 1 4 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 1 8 0 1 とテーブル 1 8 0 2 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。S 1 6 1 3 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。

【 0 2 5 2 】

図 1 8 (b) に、図 1 6 の S 1 6 1 4 で自動的に UI 部品が配置された UI 画面のキャンパスの表示例を示す。キャンパス 5 3 0 に配置された UI 部品 1 8 1 1 ~ 1 8 1 3 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 1 8 1 1 ~ 1 8 1 3 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 1 8 1 3 には自動的にアクションが設定（定義）されている。UI 部品 1 8 1 3 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 1 8 1 3 が配置された UI 画面は、UI 部品 1 8 1 3 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。

【 0 2 5 3 】

図 1 8 (c) に、ボタンである UI 部品 1 8 1 3 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 1 8 1 3 とした場合に図 8 で前述した S 8 0 1 で表示されるものである。アクションボード 9 1 0 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 1 8 (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 9 1 0 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 1 8 (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 1 8 (c) のアクションポート 9 1 0 に表示されたアクションには、図 1 6 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 S 8 0 2 ~ S 8 0 8 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。

【 0 2 5 4 】

図 1 8 (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 画面のキャンパ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、キャンパ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 1 9 の S 1 9 0 3 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 1 8 (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 1 8 (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

10

20

30

40

50

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 S 1 9 0 4 の処理によって編集可能である。すなわち、図 1 8 (c) で説明した U I 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。

【 0 2 5 5 】

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 U I 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 (U I 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 U I 部品 1 8 1 3 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 1 8 1 3 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 I D などを別の D B 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、U I 部品 1 8 1 2 (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。

【 0 2 5 6 】

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 U I 部品や U I 画面のキャンパ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。

【 0 2 5 7 】

< C R U D 生成処理 >

図 1 9 に、キャンパ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 S 7 0 3 の詳細フローチャートである。この処理は開発者用端末 1 0 0 が実行する。キャンパスのコンテキストメニューには、C R U D を生成するための選択肢が含まれる。

【 0 2 5 8 】

本実施形態では、C R U D の生成の説明のための例として、C R U D の生成前 (C R U D ボタンの配置前) に、図 2 0 (a) に示すような選択中 U I 画面のキャンパスが表示されていた場合に、キャンパスの空白領域が右クリックされてキャンパ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。C R U D 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 (C r e a t e)、読み出し (R e a d)、更新 (U p d a t e)、削除 (D e l e t e) の頭文字を繋げた語である。図 2 0 (a) のキャンパスには、U I エディタ処理において開発者によって配置された U I 部品 2 0 0 1 ~ 2 0 0 3 が表示されている。U I 部品 2 0 0 1 ~ 2 0 0 3 はいずれも種別が I n p u t の U I 部品である。U I 部品 2 0 0 1 はラベルとして「 I D」、U I 部品 2 0 0 2 は初期 V a l u e として「 N a m e」、U I 部品 2 0 0 3 はラベルとして「 a g e」、がそれぞれ表示されている。すなわち、図 2 0 (a) のキャンパスは、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 U I 画面のキャンパスである。U I 部品 2 0 0 1 は新規登録するユーザーの I D を、U I 部品 2 0 0 2 は新規登録するユーザーの名前を、U I の部品 2 0 0 3 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。U I 部品 2 0 0 1 ~ 2 0 0 3 の U I 部品 I D はそれぞれ、「 n u m b e r _ f i e l d _ I D」、「t e x t _ f i e l d _ N a m e」、「n u m b e r _ f i e l d _ a g e」であるものとする。

【 0 2 5 9 】

S 1 9 0 1では、開発者用端末 1 0 0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 2 0 (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 2 0 1 1 ~ 2 0 1 5 が表示される。

【 0 2 6 0 】

S 1 9 0 2では、開発者用端末 1 0 0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 2 0 1 3）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 2 0 1 3）が押下された場合はS 1 9 0 3に進み、そうでない場合はS 1 9 0 6に進む。

【 0 2 6 1 】

S 1 9 0 3では、開発者用端末 1 0 0は、メモリ 1 0 2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 1 8 (d)に示したものである。図 1 8 (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に表示される。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。

【 0 2 6 2 】

S 1 9 0 4では、開発者用端末 1 0 0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S 8 0 2 ~ S 8 2 3の処理と同様の処理であるため、詳細は省略する。

【 0 2 6 3 】

S 1 9 0 5では、開発者用端末 1 0 0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S 1 9 0 4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 1 9 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。

【 0 2 6 4 】

S 1 9 0 6では、開発者用端末 1 0 0、キャンバスのコンテキストメニューのうちC R U Dの生成を指示するメニュー項目（選択肢 2 0 1 5）が押下されたか否かを判定する。C R U Dの生成を指示するメニュー項目（選択肢 2 0 1 5）が押下された場合はS 1 9 0 7に進み、そうでない場合はS 1 9 1 4に進む。

【 0 2 6 5 】

S 1 9 0 7では、開発者用端末 1 0 0は、ウィザードとしてC R U Dの入力フォーム 1（ダイアログボックスである）を表示し、C R U D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 2 0 (c)に、C R U Dの入力フォーム 1の表示例を示す。領域 2 0 2 1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 2 0 2 1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 2 0 2 2に表示させることで選択を行う。

【 0 2 6 6 】

S 1 9 0 8では、開発者用端末 1 0 0では、入力フォーム 1での選択操作が行われ、N E X Tアイコン 2 0 2 3が押下されたか否かを判定する。N E X Tアイコン 2 0 2 3が押下された場合はS 1 9 0 9に進み、そうでない場合はS 1 9 0 8でN E X Tアイコン 2 0 2 3の押下を待つ。

【 0 2 6 7 】

S 1 9 0 9では、開発者用端末 1 0 0は、C R U Dの入力フォーム 2（ダイアログボックスである）を表示し、C R U D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 2 0 (d)に、C R U Dの入力フォーム 2の表示例を示す。領

10

20

30

40

50

域 2 0 3 1 には、入力フォーム 1 で選択された U I 部品の U I 部品 I D が選択肢として表示される。ユーザー（開発者）は、領域 2 0 3 1 に表示された選択肢のうち、C R U D の対象カラムとしたい選択肢を選択し、選択済み領域 2 0 3 2 に表示させることで選択を行う。

【 0 2 6 8 】

S 1 9 1 0 では、開発者用端末 1 0 0 は、入力フォーム 2 での選択操作が行われ、N E X T アイコン 2 0 3 3 が押下されたか否かを判定する。N E X T アイコン 2 0 3 3 が押下された場合は S 1 9 1 1 に進み、そうでない場合は S 1 9 1 0 で N E X T アイコン 2 0 3 3 の押下を待つ。

【 0 2 6 9 】

S 1 9 1 1 では、開発者用端末 1 0 0 は、C R U D の入力フォーム 3（ダイアログボックスである）を表示し、C R U D 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 2 0（e）に、C R U D の入力フォーム 3 の表示例を示す。領域 2 0 4 1 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 2 0 4 2 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。

【 0 2 7 0 】

S 1 9 1 2 では、開発者用端末 1 0 0 は、入力フォーム 3 での選択 / 入力操作が行われ、F I N I S H アイコン 2 0 4 3 が押下されたか否かを判定する。F I N I S H アイコン 2 0 4 3 が押下された場合は S 1 9 1 3 に進み、そうでない場合は S 1 9 1 2 で F I N I S H アイコン 2 0 4 3 の押下を待つ。

【 0 2 7 1 】

S 1 9 1 3 では、開発者用端末 1 0 0 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、C R U D ボタンを生成して選択中 U I 画面に自動的に配置する。また、C R U D 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 N e x t U I となる U I 画面を生成する。すなわち、選択中アプリに U I 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 D B セットに自動的に生成する。

【 0 2 7 2 】

S 1 9 1 4 では、開発者用端末 1 0 0 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 2 0 1 1、2 0 1 2、2 0 1 4 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 S 1 9 1 5 に進み、そうでない場合は S 1 9 1 6 に進む。

【 0 2 7 3 】

S 1 9 1 5 では、開発者用端末 1 0 0 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。

【 0 2 7 4 】

S 1 9 1 6 では、開発者用端末 1 0 0 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、S 1 9 1 7 でコンテキストメニューを非表示とし、図 1 9 の処理を終了する。閉じる操作がない場合には S 1 9 0 2 に戻る。

【 0 2 7 5 】

図 2 0（f）に、S 1 9 1 3 の処理によって自動的に配置された C R U D ボタン（自動生成部品）の表示例を示す。図 2 0（f）は、図 2 0（a）と同じ U I 画面のキャンバスの表示例であり、図 2 0（a）と同じ U I 部品には同じ符号を付す。図 2 0（f）では、図 2 0（a）に対して、U I 部品 2 0 5 1 ~ 2 0 5 3 が追加されている。この U I 部品 2 0 5 1 ~ 2 0 5 3 が C R U D ボタンである。C R U D ボタンである U I 部品 2 0 5 1 ~ 2 0 5 3 には自動的にアクションが設定されている。C R U D ボタンである U I 部品 2 0 5 1 のアクションは、U I 部品 2 0 5 1 が押下されたことに応じて、入力フォーム 3 で設定

10

20

30

40

50

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。

10

【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画面のキャンパスの表示例であり、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、キャンパ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パスのアクションは、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CRUD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。このキャンパスのアクションも、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。

20

【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。

30

【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

40

50

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。

10

20

【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一構成であるため、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。

30

40

【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一構成であるため、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 .

50

「C R U D _ T 0 1」との文字列は、処理対象となるデータベースが「s a m p l e」であり、テーブルが「C R U D _ T 0 1」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J a 10
v a S c r i p tの文字列の各行は、下記の意味である。

1行目... p a r a m s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... I D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「\$ u i . U I 部品 I D . 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J a v a S c r i p t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p a r a m sを引数として関数S Q L U p d a t e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 20
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... U p d a t eによって影響のある行(R o w)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、U p d a t eできた行があるか判定する。

9行目... 8行目が真であれば、「S u c c e s s f u l l y U p d a t e d」(更新できました)と表示する。

10行目... N e x t U IとしてUI画面I Dが「c r u d - n e w u i - U I 0 1」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p a r a m s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。 30

13行目... 12行目が真である場合に、2～5行目で定義したp a r a m sを引数として関数S Q L S a v e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、「S u c c e s s f u l l y I n s e r t e d」(追加が成功した)と表示する。 40

17行目... N e x t U IとしてUI画面I Dが「c r u d - n e w u i - U I 0 1」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、「I n s e r t i o n F a i l e d」(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。 50

・ S A V E 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 N e x t U I の U I 画面 I D の部分（ c r u d - n e w u i - U I 0 1 ）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、S Q L 関数（創作関数）を新たに作成（追加）し、書き加える。S Q L 関数を新たに作成する際には、設定欄 9 5 3 には、自動的に作成された「 S Q L S a v e 」や「 S Q L U p d a t e 」の S Q L 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】

10

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希望する C R U D 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。

20

【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】

30

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。

40

【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

50

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 S 2 2 0 8 に進む。再送指示操作がない場合に開発者用端末 1 0 0 は再送指示操作があるまで待機する。

【 0 2 9 4 】

S 2 2 0 8 では、開発者用端末 1 0 0 は、開発環境 3 0 0 に、ワンタイムパスワードの再送要求を送信する。

【 0 2 9 5 】

S 2 2 0 9 では、開発者用端末 1 0 0 は、パスワードの登録画面を開発者用端末 1 0 0 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。

【 0 2 9 6 】

S 2 2 1 0 では、開発者用端末 1 0 0 では、S 2 2 0 9 で受けつけたパスワードとしたいデータを開発環境 3 0 0 に送信する。

【 0 2 9 7 】

図 2 2 (b) に、開発環境 3 0 0 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 2 2 (a) の開発者用端末 1 0 0 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 3 0 0 側の処理である。

【 0 2 9 8 】

S 2 2 2 1 では、開発環境 3 0 0 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 1 0 0 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 2 2 (a) の S 2 2 0 2 で開発者用端末 1 0 0 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 1 0 0 から受信した場合は S 2 2 2 2 に進み、そうでない場合は受信するまで待機する。

【 0 2 9 9 】

S 2 2 2 2 では、開発環境 3 0 0 は、開発者情報 3 0 1 に行を追加し、S 2 2 2 1 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「V e r i f y」の値は「u n c o n f i r m e d」を記録する。開発者情報 3 0 1 にあるカラムは、図 1 4 (a) に示したものである。開発者情報 3 0 1 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。

【 0 3 0 0 】

S 2 2 2 3 では、開発環境 3 0 0 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、S 2 2 2 1 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 3 0 0 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。

【 0 3 0 1 】

S 2 2 2 4 では、開発環境 3 0 0 は、開発者用端末 1 0 0 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 2 2 (a) の S 2 2 0 4 で開発者用端末 1 0 0 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 S 2 2 2 5 に進み、そうでない場合には S 2 2 2 4 で待機する。

【 0 3 0 2 】

S 2 2 2 5 では、開発環境 3 0 0 は、外部のワンタイムパスワード発行システムに対して、S 2 2 2 5 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。

【 0 3 0 3 】

S 2 2 2 6 では、開発環境 3 0 0 は、外部のワンタイムパスワード発行システムから、

10

20

30

40

50

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S 2 2 2 8に進み、そうでない場合にはS 2 2 7へ進む。

【0304】

S 2 2 2 7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S 2 2 0 8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S 2 2 2 3へ進む、そうでない場合はS 2 2 2 7で待機する。

【0305】

S 2 2 2 8では、開発環境300は、S 2 2 2 2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、S 2 2 2 2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。

10

【0306】

S 2 2 2 9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S 2 2 3 0に進み、有効なアカウント数が閾値2以下であればS 2 2 3 2に進む。

20

【0307】

S 2 2 3 0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S 2 2 3 2に進み、追加済みでなければS 2 2 3 1に進む。

【0308】

S 2 2 3 1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。

30

【0309】

S 2 2 3 2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S 2 2 3 3に進み、有効なアカウント数が閾値1以下であればS 2 2 3 6に進む。

40

【0310】

S 2 2 3 3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、S 2 2 3 1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みであるか否かを判定する。更新済みである場合にはS 2 2 3 6へ進む、そうでない場合にはS 2 2 3 4へ進む。

【0311】

S 2 2 3 4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

50

スタンス名を、S 2 2 3 1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」を割り当てていたのが、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増えると、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、S 2 2 3 3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。

10

【0312】

S 2 2 3 5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、S 2 2 3 1の処理とS 2 2 3 4の処理が行われる。

20

【0313】

S 2 2 3 6では、開発環境300は、S 2 2 2 2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り当てる。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。S 2 2 3 2 ~ S 2 2 3 5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。

30

【0314】

S 2 2 3 7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S 2 2 0 5において開発者用端末100によって受信される。

【0315】

S 2 2 3 8では、開発環境300は、前述したS 2 2 1 0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ことを示している。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、DB max

40

50

(2 3 1 7) が開発者用端末 1 0 0 0 0 個目の D B であり、それ以降は別の D B インスタンスに D B が作られる。発明者毎の D B の中にはさらに 1 以上のテーブルが含まれる。例えば D B 1 (2 3 1 4) 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 D B 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。

【 0 3 1 7 】

S 2 2 3 1 の処理によって、D B インスタンス 1 の D B の数が上限 (閾値 1) に達する前に、次の D B インスタンス 2 が生成される。そして、D B インスタンス 1 に D B m a x (2 3 1 7) 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 3 0 1 に追加登録された場合、S 2 2 3 2 で Y e s となり、追加登録された開発者は D B インスタンス 2 (2 3 2 0) に接続するように開発者情報 3 0 1 に登録される。そして、D B インスタンス 2 (2 3 2 0) にそれ以降追加されたアカウント (開発者) 用の D B が生成されていく。なお、D B インスタンス 2 (2 3 2 0) に入る D B 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 D B インスタンスに接続するように制御される。

10

【 0 3 1 8 】

図 2 3 (b) に、シングルテナント実行環境の D B セット 4 5 7 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、D B セット 4 5 7 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 D B の増加、アクセスの集中が起こる可能性は低い。従って、D B セット 4 5 7 には、D B 環境としては D B インスタンス 1 (2 3 5 0) の 1 つだけが用意される。また、開発者情報 D B 2 3 5 1 には 1 アカウント文の情報 (例えば開発者 A の D B 情報 2 3 5 2) だけが記録される。格納される D B の数にも上限となる閾値は用意されない。また、図 2 2 (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。

20

【 0 3 1 9 】

< デプロイ処理 >

図 2 4 (a) に、開発者用端末 1 0 0 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 1 0 0 が実行する処理であり、図 4 の S 4 2 6 の詳細フローチャートである。

30

【 0 3 2 0 】

図 2 5 (a) に、選択中アプリがモバイル用アプリである場合の U I エディタ画面の表示例を示す。キャンバス 2 5 0 1 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 5 0 6 が押下されると図 2 4 (a) の処理が開始される。

【 0 3 2 1 】

S 2 4 0 1 では、開発者用端末 1 0 0 は、開発環境 3 0 0 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 1 0 0 のメモリ 1 0 2 に保持された最新の定義情報を開発環境 3 0 0 に保存する保存処理を行って

40

【 0 3 2 2 】

S 2 4 0 2 では、開発者用端末 1 0 0 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。

【 0 3 2 3 】

S 2 4 0 3 では、開発者用端末 1 0 0 は、開発環境 3 0 0 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 S 2 4 0 4 に進み、そうでない場合には S 2 4 0 5 へ進む。

【 0 3 2 4 】

S 2 4 0 4 では、開発者用端末 1 0 0 は、デプロイが失敗した旨のエラー表示を行う。

50

デプロイが失敗した場合は、後述の S 2 4 0 7、S 2 4 0 9 の処理は行われない。

【 0 3 2 5 】

S 2 4 0 5 では、開発者用端末 1 0 0 は、開発環境 3 0 0 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 (U R L) が含まれる。成功通知を受信した場合には S 2 4 0 6 に進み、そうでない場合には S 2 4 0 3 へ進む。

【 0 3 2 6 】

S 2 4 0 6 では、開発者用端末 1 0 0 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 S 2 4 0 8 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 S 2 4 0 7 に進む。モバイル用でない 10
場合には後述する S 2 4 0 9 の Q R コード (登録商標) の表示は行わない。

【 0 3 2 7 】

S 2 4 0 7 では、開発者用端末 1 0 0 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 (例えばキャンパスを含む U I 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 U R L (実行環境にアクセスする U R L) 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 U I 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・ 20
検証することができる。この S 2 4 0 7 の処理によって、デプロイ済みのアプリの U R L (実行環境にアクセスする U R L) 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 3 3 (c) または図 3 5 (c) のフローチャートを用いて説明する。

【 0 3 2 8 】

S 2 4 0 8 では、開発者用端末 1 0 0 は、成功通知に含まれるアプリの U R L を Q R コード化する処理を行う。すなわち、アプリの U R L の情報を含む Q R コード (2 次元コード) を生成する。

【 0 3 2 9 】

S 2 4 0 9 では、開発者用端末 1 0 0 は、S 2 4 0 9 で生成した Q R コードを表示する。Q R 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 2 5 (b) に、S 2 4 0 9 での Q R コードの表示例を示す。図 2 5 (b) は、図 2 5 (a) の状態からデプロイボタン 5 0 6 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パスに重畳して、Q R コードダイアログ 2 5 1 0 が表示される。Q R コードダイアログ 2 5 1 0 には、S 2 4 0 8 で生成した Q R コード 2 5 1 1 と、閉じるボタン 2 5 1 2 と、新しいタブで開くボタン 2 5 1 3 が表示される。なお、Q R コード 2 5 1 1 が有する情報であるアプリの U R L は、U I 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 U R L 2 5 0 4 (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 U R L であって、開発者用端末 1 0 0 にアクセスする U R L) とは別の U R L である。Q R コード 2 5 1 1 は、デプロイが完了したことによって表 30
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。 40

【 0 3 3 0 】

開発者 (ユーザー) は、開発者用端末 1 0 0 のディスプレイ 1 0 5 に表示された Q R コード 2 5 1 1 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 U R L (実行環境にアクセスする U R L) 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 U R L 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 2 0 1 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 U I が表示され、手持ちのス 50

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】

10

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】

20

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】

30

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。

40

【0337】

S2424では、開発環境 300 は、選択中実行環境であるシングルテ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パ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テナント実行環境に送信し、パ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

50

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。

【 0 3 3 9 】

S 2 4 2 6 では、開発環境 3 0 0 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 1 0 0 に送信する。

【 0 3 4 0 】

S 2 4 2 7 では、開発環境 3 0 0 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 S 2 4 2 8 に進み、そうでない場合には S 2 4 2 5 に進む。

【 0 3 4 1 】

S 2 4 2 8 では、開発環境 3 0 0 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 1 0 0 に送信する。ここで送信した成功通知が、前述の図 2 4（a）の S 2 4 0 5 で開発者用端末 1 0 0 に受信される。

【 0 3 4 2 】

< U I 画面のテンプレート >

U I 画面のテンプレートについて説明する。U I 画面には、ユーザーが作成した U I 画面とは別に、テンプレートとして予め用意されている U I 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 U I 画面に過去に U I 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、U I 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 U I エディタ処理によって編集が可能である。

【 0 3 4 3 】

図 2 6（a）に、サブメニュー領域 5 2 0 に U I 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 S 4 0 1 の処理によって行われるものである。図 2 6（a）は、複数の種別に分類された U I 画面のうち、A u t h e n t i c a t i o n U I（認証用の U I 画面、認証用画面）のグループに分類される U I 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 2 6 1 0 は展開して表示しているグループが A u t h e n t i c a t i o n U I であることを示している。グループ名 2 6 1 0 が押下されると A u t h e n t i c a t i o n U I 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 5 2 0 に表示される。選択肢 2 6 1 1 ~ 2 6 2 2 がそれぞれ 1 つの U I 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。

【 0 3 4 4 】

図 2 6（b）に、選択肢 2 6 1 1 を選択して、選択肢 2 6 1 1 に対応するテンプレート画面（U I 画面）である「S i g n i n I D」をキャンバス 2 6 3 0 に表示した場合の表示例を示す。図 2 6（b）の画面は、選択肢 2 6 1 1 が押下されたことに応じて、前述した図 4 の S 4 0 4 で表示される。キャンバス 2 6 3 0 には、雛形部品 2 6 3 1 ~ 2 6 3 4 が表示されている。雛形部品 2 6 3 1 ~ 2 6 3 4 は、ユーザー（開発者）がサブメニュー領域 5 2 0 の U I 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 U I 部品である。雛形部品 2 6 3 1 ~ 2 6 3 4 はそれぞれ以下のような U I 部品である。

・雛形部品 2 6 3 1：O u t p u t F i e l d で、「S i g n I n」と表示するための U I 部品。

・雛形部品 2 6 3 2：I n p u t の種別の T e x t F i e l d 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 U I 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 U I 部品。

・雛形部品 2 6 3 3：ボタンの U I 部品で、「C R E A T E A C C O U N T？」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「S i g n U p f o r m」の画面（図 2 7（c）に示す画面で、選択肢 2 6 1 4 に

10

20

30

40

50

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、テンプレー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、テンプレート画面の一種である「Reset Password Step 1」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 for 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 2 7 (j) : 選択肢 2 6 2 1 に対応する U I 画面。
- ・図 2 7 (k) : 選択肢 2 6 2 2 に対応する U I 画面。

【 0 3 4 8 】

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 S 4 0 5 以降の処理で説明した通り、U I 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 U I 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パ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 U I 部品（雛形部品ではない U I 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 U I 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 2 6 (b) の「 S i g n i n I D 」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 U I 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 O K となっていない状態では遷移先の U I 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 U I 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。

【 0 3 4 9 】

図 2 6 (c) に、テンプレート画面ではない U I 画面のキャンパスに配置されたボタン 2 6 5 4 のコンテキストメニュー 2 6 5 5 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 S 7 1 1 ~ S 7 2 0 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。

【 0 3 5 0 】

図 2 6 (d) に、テンプレート画面のキャンパスに配置されたボタン 2 6 3 4 のコンテキストメニュー 2 6 3 5 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 S 7 2 1 ~ S 7 2 4 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。

【 0 3 5 1 】

図 2 6 (e) に、図 2 6 (b) に示したテンプレート画面のキャンパスに対して編集（カスタマイズ）を加えた後の表示例を示す。図 2 6 (b) と図 2 6 (e) とで同じ U I 部品には同じ符号を付す。U I 部品 2 6 4 1 は、開発者によって追加された出力の種別の U I 部品であり、図 1 2 の S 1 2 1 7 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。U I 部品 2 6 4 2 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 U I 部品である。また、この画面のキャンパ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 U I 部品 2 6 4 2 に表示する、というアクションを、キャンパ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、社内で行き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になってしまう。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 :

10

20

30

40

50

```
# C 2 D 2 E 9 , month : Mar } , { id : Apr , value : 34.25 , color : # 9 A 8 B A 5 , month : Apr } , { id : May , value : 24.65 , color : # E 3 C 5 D 5 , 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 : # 3 9 4 E 7 9 , 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2811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部品またはキャンパ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

10

20

30

40

50

ョンボード 2900 に表示された文字列 (JavaScript 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】

10

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 CHART 2803 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。

20

【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、パンくずリスト (Breadcrumb) 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。

30

【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。

40

【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。

50

【 0 3 6 6 】

S 3 0 0 3 では、開発者用端末 1 0 0 は、コンテキストメニュー 3 1 2 0 に含まれる選択肢のうち、アクション 3 1 2 2 が押下（選択）されたか否かを判定する。アクション 3 1 2 2 が押下（選択）された場合は S 3 0 2 0 に進み、そうでない場合には S 3 0 0 4 へ進む。

【 0 3 6 7 】

S 3 0 0 4 では、開発者用端末 1 0 0 は、コンテキストメニュー 3 1 2 0 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 S 3 0 0 5 に進み、そうでない場合には S 3 0 0 6 へ進む。

【 0 3 6 8 】

S 3 0 0 5 では、開発者用端末 1 0 0 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。

【 0 3 6 9 】

S 3 0 0 6 では、開発者用端末 1 0 0 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、S 3 0 0 7 でコンテキストメニューを非表示とし、図 3 0 の処理を終了する。閉じる操作がない場合には S 3 0 0 2 に戻る。

【 0 3 7 0 】

S 3 0 1 0 では、開発者用端末 1 0 0 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 3 1 (b) ~ 図 3 1 (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 3 1 (b) ~ 図 3 1 (f) において、同じ表示アイテムには同じ符号を付す。図 3 1 (b) の適用ボタン 3 1 3 1 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 3 1 3 1 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 3 1 3 2 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 3 1 3 3 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。

【 0 3 7 1 】

S 3 0 1 1 では、開発者用端末 1 0 0 は、プロパティボックス内のカラム選択欄 3 1 3 2 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 S 3 0 1 2 に進み、そうでない場合には S 3 0 1 3 に進む。なお、カラム選択欄 3 1 3 2 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、「Data Grid」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「Data Grid」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）。

【 0 3 7 2 】

S 3 0 1 2 では、開発者用端末 1 0 0 は、プロパティボックスのうちカラム選択欄 3 1 3 2 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 3 1 (c) に、カラム選択欄 3 1 3 2 のうち、タブ 3 1 3 2 E（「Company E」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 3 1 3 2 より下側の領域には、タブ 3 1 3 2 E に対応するカラムの設定を受け付ける画面として、タブ 3 1 3 2 E の設定画面の上部領域 3 1 3 4 t 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

10

20

30

40

50

3 1 3 5 が含まれる。タブ 3 1 3 2 E 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 3 1 (d) に、スクロールさせて、タブ 3 1 3 2 E の設定画面の下部領域 3 1 3 4 d を表示させた場合の表示例を示す。下部領域 3 1 3 4 d には、タブ 3 1 3 2 E 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 3 1 3 6 と、タブ 3 1 3 2 E 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 3 1 3 7 とが含まれる。カラム選択欄 3 1 3 2 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 3 1 3 4 t 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、S 3 0 1 1 で「Data Grid」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 3 1 3 2 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。

10

【 0 3 7 3 】

S 3 0 1 3 では、開発者用端末 1 0 0 は、プロパティボックス内のカラム追加ボタン 3 1 3 3 が押下されたか否かを判定する。カラム追加ボタン 3 1 3 3 が押下された場合には S 3 0 1 4 に進み、そうでない場合には S 3 0 1 5 に進む。

【 0 3 7 4 】

S 3 0 1 4 では、開発者用端末 1 0 0 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 3 1 (e) に示すカラムの追加フォーム 3 1 3 8 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 3 1 3 2 に新しいカラムに対応するタブが追加される。

20

【 0 3 7 5 】

図 3 1 (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 3 1 (c) と比較して、カラム選択欄 3 1 3 2 には、新しいカラムに対応するタブ 3 0 3 2 F (「Company F」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 3 1 3 2 より下側の領域には、追加したカラムに対応するタブ 3 0 3 2 F の設定画面の上部領域 3 1 3 9 t 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。

【 0 3 7 6 】

S 3 0 1 5 では、開発者用端末 1 0 0 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 3 1 3 2 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 S 3 0 1 6 へ進み、そうでない場合は S 3 0 1 7 へ進む。

30

【 0 3 7 7 】

S 3 0 1 6 では、開発者用端末 1 0 0 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 3 1 3 5 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、Text Input、Number Input、Combo Box、Multi Select、CheckBox、Data Picker、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 Text Input 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）、Number Format、Data Format、Options、Footer、Footer Text、Sorting、Visibility。また、「Data Grid」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）、Data Grid 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

40

50

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。

10

【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。

20

【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。

30

40

【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かを判定する。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

50

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 3 2 2 0 で「DATA GRID」が選択されており、アクションボード 3 2 1 1 a 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 3 2 1 1 a 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、前述したプロパディボックスの「Data Grid」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 3 2 1 1 a 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。

10

【0386】

図 3 2 (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 3 2 1 2）に対応するアクションボードの表示例を示す。選択領域 3 2 2 0 で「MONTH」が選択されており、アクションボード 3 2 1 2 a がラベルまたはカラム ID が「MONTH」のカラム（カラム 3 1 1 1）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 3 2 1 2 a 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 3 2 1 2 a に例えば、アクションボード 3 2 1 2 a 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 3 2 1 2 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。

20

30

【0387】

なお、選択領域 3 2 2 0 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】

S 3 0 2 3 では、開発者用端末 1 0 0 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 S 8 0 2 ~ S 8 2 3 の処理と同様である。

40

【0389】

S 3 0 2 4 では、開発者用端末 1 0 0 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 S 3 0 2 3 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 3 2 1 0 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

50

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。

10

【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。

20

【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とする。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部品やキャンパ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。

30

【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。

40

【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341

50

2 から、実行環境用プログラムを生成して、ストレージ 3 2 0 のうちログイン開発者用の領域に保存する。すなわち、図 3 4 の UI 定義情報 3 4 1 1 から抽出したアクション記述部分 3 4 1 2 に基づいて、実行環境用プログラム 3 4 1 3 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 3 4 1 3 は J a v a S c r i p t 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 3 4 1 3 は、アクション記述部分 3 4 1 2 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。

【 0 3 9 8 】

図 3 3 (a) で説明した保存処理によって開発環境 3 0 0 に保存された UI 定義情報 3 4 1 1 と実行環境用プログラム 3 4 1 3 が、デプロイ処理によって実行環境 4 0 0 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 2 4 の S 2 4 2 3 または S 2 4 2 4 で行われる。これによって、図 3 4 に図示した通り、実行環境 4 0 0 に UI 定義情報 3 4 2 1 と実行環境用プログラム 3 4 2 3 が記録される。実行環境 4 0 0 に UI 定義情報 3 4 2 1 と実行環境用プログラム 3 4 2 3 はそれぞれ、UI 定義情報 3 4 1 1 と実行環境用プログラム 3 4 1 3 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。

10

【 0 3 9 9 】

図 3 3 (b) に、実行環境 4 0 0 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 4 0 0 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 2 0 0 または 2 0 1 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。

20

【 0 4 0 0 】

図 3 3 (c) に、アプリユーザー用端末 2 0 0 または 2 0 1 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、アプリユーザー用端末 2 0 0 または 2 0 1 の C P U 1 0 1 が実行する処理であり、アプリユーザー用端末 2 0 0 または 2 0 1 のブラウザソフトで、実行環境 4 0 0 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 3 3 (b) の実行環境 4 0 0 におけるアプリ実行処理と、図 3 3 (c) のアプリユーザー用端末 2 0 0 または 2 0 1 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 2 0 0 または 2 0 1 による処理について、代表してアプリユーザー用端末 2 0 0 が実行するものとして説明する (アプリユーザー用端末 2 0 1 に関する説明は同様であるため省略する) 。

30

【 0 4 0 1 】

図 3 3 (b) の S 3 3 1 1 では、実行環境 4 0 0 は、アプリユーザー用端末 2 0 0 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 3 3 (c) の S 3 3 3 3 でアプリユーザー用端末 2 0 0 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 S 3 3 1 2 に進み、そうでない場合は S 3 3 1 1 で待機する。

【 0 4 0 2 】

S 3 3 1 2 では、実行環境 4 0 0 は、UI 定義情報をアプリユーザー用端末 2 0 0 に送信する。これによって、図 3 4 に図示した通り、実行環境 4 0 0 に記録された UI 定義情報 3 4 2 1 がアプリユーザー用端末 2 0 0 にダウンロードされ、UI 定義情報 3 4 3 1 として記録される。UI 定義情報 3 4 2 1 と UI 定義情報 3 4 3 1 とは同じ情報である。

40

【 0 4 0 3 】

S 3 3 1 3 では、実行環境 4 0 0 は、アプリユーザー用端末 2 0 0 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 3 3 (c) の S 3 3 4 5 でアプリユーザー用端末 2 0 0 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 S 3 3 1 4 に進み、そうでない場合には S 3 3 2 0 に進む。

【 0 4 0 4 】

S 3 3 1 4 では、実行環境 4 0 0 は、アプリユーザー用端末 2 0 0 から入力項目の値を

50

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。

10

【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。

20

【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。

30

【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。

40

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

50

たか否かを判定する。実行環境 4 0 0 は、アプリユーザー用端末 2 0 0 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 4 1 5、4 5 5、4 6 5、4 7 5 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 4 2 2、4 5 6 c、4 6 6 c、4 7 6 c 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 2 0 0 に送信する。S 3 3 3 2 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 S 3 3 3 3 に進み、そうでない場合には S 3 3 3 2 でクライアント用プログラムの受信を待つ。

【 0 4 1 2 】

S 3 3 3 3 では、アプリユーザー用端末 2 0 0 は、実行環境 4 0 0 に UI 定義情報の取得要求を送信する。S 3 3 3 2 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。S 3 3 3 3 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 3 3 (b) の S 3 3 1 1 において実行環境 4 0 0 に受信される。

【 0 4 1 3 】

S 3 3 3 4 では、アプリユーザー用端末 2 0 0 は、実行環境 4 0 0 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 3 3 (b) の S 3 3 1 2 で実行環境 4 0 0 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 1 0 2 に記録して S 3 3 3 5 に進み、そうでない場合には S 3 3 3 4 で UI 定義情報の受信を待つ。UI 定義情報はメモリ 1 0 2 (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 2 0 0 は、メモリ 1 0 2 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 1 0 5 に表示する。

【 0 4 1 4 】

S 3 3 3 5 では、アプリユーザー用端末 2 0 0 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 (遷移先のキャンパ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 S 3 3 3 6 へ進み、そうでない場合には S 3 3 5 0 へ進む。

【 0 4 1 5 】

S 3 3 3 6 では、アプリユーザー用端末 2 0 0 は、メモリ 1 0 2 に記録された UI 定義情報 (図 3 4 における UI 定義情報 3 4 3 1) のうち、S 3 3 3 5 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。

【 0 4 1 6 】

S 3 3 3 7 では、アプリユーザー用端末 2 0 0 は、S 3 3 3 6 で抽出した記述部分から、半角の「\$ u i」という文字列を先頭から検索する。

【 0 4 1 7 】

S 3 3 3 8 では、アプリユーザー用端末 2 0 0 は、S 3 3 3 7 の検索の結果、「\$ u i」という文字列があったか否かを判定する。「\$ u i」があった場合には S 3 3 3 9 に進み、そうでない場合には S 3 3 4 3 に進む。

【 0 4 1 8 】

S 3 3 3 9 には、アプリユーザー用端末 2 0 0 は、S 3 3 3 8 で見つかった「\$ u i」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「\$ u i」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。「\$

10

20

30

40

50

u i」が左辺にある場合には S 3 3 4 0 に進み、そうでない場合には S 3 3 4 1 に進む。

【 0 4 1 9 】

S 3 3 4 0 では、アプリユーザー用端末 2 0 0 は、S 3 3 3 8 で見つかった「\$ u i」で始まる所定の構造の要素文字列（\$ u i . U I 部品 I D . 対象となる情報の種別）に基づく情報を、メモリ 1 0 2 に生成した項目定義リスト（図 3 4 の項目定義リスト 3 4 3 3）に出力項目として記録する。具体的には、「\$ u i」で始まる「.」（ピリオド）で区切られた構造の要素文字列（\$ u i . U I 部品 I D . 対象となる情報の種別）から U I 部品 I D（項目コード、コンポーネント識別子）を取得し、その U I 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「\$ u i」が見つかり、それが左辺にあり、U I 部品 I D が「U I 部品 I D 3」であった場合には、図 3 4 の項目定義リスト 3 4 3 3 のように、出力項目に U I 部品 I D 3 がある旨が記録される。なお、項目定義リスト 3 4 3 3 はワークメモリであるメモリ 1 0 2 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 S 3 3 4 8 で N o または S 3 3 4 9 の処理を終了）、アプリを終了したこと（S 3 3 5 0 で Y e s）に応じて、自動的に消去するものとする。

【 0 4 2 0 】

S 3 3 4 1 では、アプリユーザー用端末 2 0 0 は、S 3 3 3 8 で見つかった「\$ u i」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 S 3 3 4 2 へ進み、そうでない場合には S 3 3 3 7 へ進む。なお、S 3 3 3 9 で N o と判定された場合は、「\$ u i」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、S 3 3 4 1 の判定をせずに S 3 3 4 2 へ進むものとしてもよい。

【 0 4 2 1 】

S 3 3 4 2 では、アプリユーザー用端末 2 0 0 は、S 3 3 3 8 で見つかった「\$ u i」で始まる所定の構造の要素文字列（\$ u i . U I 部品 I D . 対象となる情報の種別）に基づく情報を、メモリ 1 0 2 に生成した項目定義リスト（図 3 4 の項目定義リスト 3 4 3 3）に出力項目として記録する。具体的には、「\$ u i」で始まる所定の構造の要素文字列（\$ u i . U I 部品 I D . 対象となる情報の種別）から U I 部品 I D（項目コード、コンポーネント識別子）を取得し、その U I 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「\$ u i」が見つかり、それが右にあり、U I 部品 I D が「U I 部品 I D 1」であった場合には、図 3 4 の項目定義リスト 3 4 3 3 のように、入力項目に U I 部品 I D 1 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「\$ u i」が見つかり、それが「=」のない文にあり、U I 部品 I D が「U I 部品 I D 2」であった場合には、図 3 4 の項目定義リスト 3 4 3 3 のように、入力項目に U I 部品 I D 2 がある旨が記録される。

【 0 4 2 2 】

S 3 3 4 2 の処理を終えると、S 3 3 3 7 に進み、S 3 3 3 6 で抽出した記述部分の続きの部分において、さらに「\$ u i」の文字列を検索し、S 3 3 3 8 で「\$ u i」が無いと判定されるまで S 3 3 3 9 ~ S 3 3 4 2 の処理を繰り返す。すなわち、S 3 3 3 6 で抽出した記述部分に含まれる全ての \$ u i の文字列について、S 3 3 3 9 ~ S 3 3 4 2 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 3 4 3 3 に記録する。

【 0 4 2 3 】

S 3 3 4 3 では、アプリユーザー用端末 2 0 0 は、S 3 3 3 5 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 3 4 3 3 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、S 3 3 3 6 で抽出した記述部分に「\$ u i . U I 部品 I D 1 . v a l u e」と記載されていた場合、U I 部品 I D 1 の U I 部品内の「v a l u e」という種別の情報に値があるかどうかを判定する。U I 部品 I D 1 の U I 部品が T e x t F i e l d であった場合には、v a l u e は、その T e x t F i e l d

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 `Text Field` である `UI 部品 ID 1` 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 3 4 3 3 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 `S 3 3 4 4` へ進み、そうでない場合には `S 3 3 4 5` へ進む。

【 0 4 2 4 】

`S 3 3 4 4` では、アプリユーザー用端末 2 0 0 は、入力項目の値を含む、`S 3 3 3 5` 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 2 0 0 のディスプレイ 1 0 5 に表示されるアプリの画面に表示された `Text Field` である `UI 部品 ID 1` 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。

10

【 0 4 2 5 】

`S 3 3 4 5` では、アプリユーザー用端末 2 0 0 は、`S 3 3 3 5` 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 4 0 0 に送信する。`S 3 3 4 4` 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 4 0 0 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 2 0 0 のディスプレイ 1 0 5 に表示されるアプリの画面に表示された `Text Field` である `UI 部品 ID 1` 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 4 0 0 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 4 0 0 で実行される前述した図 3 3 (b) の `S 3 3 1 3` で実行環境 4 0 0 に受信され、`S 3 3 1 4` ~ `S 3 3 1 9` で説明した通り、`S 3 3 3 5` で検知したトリガーに応じたアクションが実行環境 4 0 0 で実行される。

20

【 0 4 2 6 】

`S 3 3 4 6` では、アプリユーザー用端末 2 0 0 は、`S 3 3 4 5` 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 4 0 0 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 `S 3 3 4 7` へ進み、そうでない場合には `S 3 3 4 6` でアクションの結果の受信を待つ。

【 0 4 2 7 】

`S 3 3 4 7` では、アプリユーザー用端末 2 0 0 は、`S 3 3 4 6` で受信したアクションの結果を反映してディスプレイ 1 0 5 に表示されたアプリの画面を更新する。この時もメモリ 1 0 2 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。

30

【 0 4 2 8 】

`S 3 3 4 8` では、アプリユーザー用端末 2 0 0 は、`S 3 3 4 6` 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かを判定する。出力項目の値が含まれていた場合には `S 3 3 4 9` に進み、そうでない場合には `S 3 3 5 0` へ進む。

【 0 4 2 9 】

`S 3 3 4 9` では、アプリユーザー用端末 2 0 0 は、`S 3 3 4 6` 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 1 0 2 に保持した項目定義リスト 3 4 3 3 に記録した出力項目の情報に基づいて、ディスプレイ 1 0 5 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。

40

【 0 4 3 0 】

`S 3 3 5 0` では、アプリユーザー用端末 2 0 0 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 2 0 0 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 `S 3 3 3 5` に進み、アプリを終了させるイベントがあ

50

った場合には、アプリの画面の非表示として図 3 3 (c) の処理を終了する。

【 0 4 3 1 】

図 1 0 (d) に示したアクションボード 9 1 0 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、S 3 3 3 8 ~ S 3 3 4 2 で説明した「\$ u i」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。

【 0 4 3 2 】

2 行目には「const user id = \$ u i . t e x t _ f i e l d _ a . v a l u e ;」と記載されている。この文において、「\$ u i」が右辺にあるため、S 3 3 4 2 の処理で、項目定義リストに「t e x t _ f i e l d _ a」が入力項目として記録される。すなわち「user id」という変数の値に、アプリの画面において「t e x t _ f i e l d _ a」にユーザーが入力した値が取得されて設定される。

10

【 0 4 3 3 】

4 行目の文において、「\$ u i」を含む「\$ u i . t e x t _ a r e a _ a . v a l u e」との文字列が左辺にあるため、S 3 3 4 0 の処理で、項目定義リストに「t e x t _ a r e a _ a」が出力項目として記録される。すなわち「t e x t _ a r e a _ 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、「t e x t _ a r e a _ 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、「\$ u i . t e x t _ a r e a _ a . v a l u e」が左辺にあるため出力項目として扱われる。

20

【 0 4 3 4 】

また、以下に、「=」の無い文に「\$ u i」があるアクションの文字列の例を示す

【 0 4 3 5 】

（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  S Q L S a v e ( {  
2      P R E F _ C O D E :   $ u i . P R E F _ C O D E . v a l u e ,  
3      P R E F _ N A M E :   $ u i . P R E F _ N A M E . v a l u e  
4  } ) ;  
5  / /   画面遷移  
6  $ f n . n e x t U I (   L i s t _ i p _ d e m o   ) ;
```

30

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（`\$ u i . P R E F _ C O D E . v a l u e` , ` \$ u i . P R E F _ N A M E . v a l u e`）を S Q L S a v e 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、S Q L S a v e 関数のパラメータとして使用される \$ u i . { 部品 I D } . { 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。

【 0 4 3 6 】

40

また、「\$ u i」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、v a l u e 以外でもよく、例えば色でも良い。例えば、「\$ u i . U I 部品 I D . c o l o r = # 0 0 8 0 0 0 ;」と記述することで、UI 部品の色を変更することができる。

【 0 4 3 7 】

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、「\$ u i」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 部品を入力項目として扱うか、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

50

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、入力項目として扱うか、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行うことができる。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連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で実行する処理に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な情報とアプリユーザー用端末200で実行する処理に必要な情報とを、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三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

10

20

30

40

50

分 3 4 1 2 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。

【 0 4 4 9 】

S 3 5 0 7 では、開発環境 3 0 0 は、S 3 5 0 6 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。

【 0 4 5 0 】

S 3 5 0 8 では、開発環境 3 0 0 は、S 3 3 3 7 の検索の結果、「\$ u i」という文字列があったか否かを判定する。「\$ u i」があった場合には S 3 5 0 9 に進み、そうでない場合には処理を終了する。S 3 5 0 8 で No となるのは、全ての「\$ u i」の検索が終わったか、アクション記述部分に全く「\$ u i」が無かった場合である。

10

【 0 4 5 1 】

S 3 5 0 9 では、開発環境 3 0 0 は、図 3 3 (c) の S 3 3 3 9 と同様、S 3 5 0 8 で見つかった「\$ u i」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。「\$ u i」が左辺にある場合には S 3 5 1 0 に進み、そうでない場合には S 3 5 1 1 に進む。

【 0 4 5 2 】

S 3 5 1 0 では、開発環境 3 0 0 は、S 3 3 3 8 で見つかった「\$ u i」で始まる所定の構造の要素文字列（\$ u i . UI 部品 ID . 対象となる情報の種別）に基づく情報を、S 3 5 0 5 で生成したクライアント用 UI 定義情報 3 6 1 5 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 3 6 1 5 のうち、出力項目を定義する配列に、S 3 3 3 8 で見つかった「\$ u i」で始まる所定の構造の要素文字列（\$ u i . 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、“ u i I t e m s O u t ”の配列に記録する。例えば UI 部品 ID が「 R E S U L T 」であることを示す「\$ u i」が左辺に見つかった場合、図 3 6 に図示したように記録される。

20

【 0 4 5 3 】

S 3 5 1 1 では、開発環境 3 0 0 は、図 3 3 (c) の S 3 3 4 1 と同様、S 3 5 0 8 で見つかった「\$ u i」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 S 3 5 1 2 へ進み、そうでない場合には S 3 5 0 7 へ進む。なお、S 3 5 0 9 で No と判定された場合は、「\$ u i」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、S 3 5 1 1 の判定をせずに S 3 5 1 2 へ進むものとしてもよい。

30

【 0 4 5 4 】

S 3 5 1 2 では、開発環境 3 0 0 は、S 3 3 3 8 で見つかった「\$ u i」で始まる所定の構造の要素文字列（\$ u i . UI 部品 ID . 対象となる情報の種別）に基づく情報を、S 3 5 0 5 で生成したクライアント用 UI 定義情報 3 6 1 5 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 3 6 1 5 のうち、入力項目を定義する配列に、S 3 3 3 8 で見つかった「\$ u i」で始まる所定の構造の要素文字列（\$ u i . 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、“ u i I t e m s I n ”の配列に記録する。例えば UI 部品 ID が「 I N P U T _ 1 」であることを示す「\$ u i」が右辺に見つかった場合、図 3 6 に図示したように記録される。また、例えば UI 部品 ID が「 I N P U T _ 2 」であることを示す「\$ u i」が「 = 」のない文に見つかった場合、図 3 6 に図示したように記録される。

40

【 0 4 5 5 】

S 3 5 1 2 の処理を終えると、S 3 5 0 7 に進み、S 3 5 0 6 で抽出した記述部分の続きの部分において、さらに「\$ u i」の文字列を検索し、S 3 5 0 8 で「\$ u i」が無いと判定されるまで S 3 5 0 7 ~ S 3 5 1 2 の処理を繰り返す。すなわち、S 3 5 0 6 で抽出した記述部分に含まれる全ての \$ u i の文字列について、S 3 5 0 9 ~ S 3 5 1 2 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 3 6 1 5 に記録する。

50

【 0 4 5 6 】

図 3 5 (b) に、実行環境 4 0 0 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 4 0 0 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 2 0 0 または 2 0 1 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。

【 0 4 5 7 】

図 3 3 (c) に、アプリユーザー用端末 2 0 0 または 2 0 1 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、アプリユーザー用端末 2 0 0 または 2 0 1 の C P U 1 0 1 が実行する処理であり、アプリユーザー用端末 2 0 0 または 2 0 1 のブラウザソフトで、実行環境 4 0 0 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 3 5 (b) の実行環境 4 0 0 におけるアプリ実行処理と、図 3 5 (c) のアプリユーザー用端末 2 0 0 または 2 0 1 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 2 0 0 または 2 0 1 による処理について、代表してアプリユーザー用端末 2 0 0 が実行するものとして説明する（アプリユーザー用端末 2 0 1 に関する説明は同様であるため省略する）。

10

【 0 4 5 8 】

図 3 5 (b) の S 3 5 2 1 では、実行環境 4 0 0 は、アプリユーザー用端末 2 0 0 から送信された U I 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 U I 定義情報の取得要求は、図 3 5 (c) の S 3 5 4 3 でアプリユーザー用端末 2 0 0 から送信されるものである。U I 定義情報の取得要求を受信した場合は S 3 5 2 2 に進み、そうでない場合は S 3 5 2 1 で待機する。

20

【 0 4 5 9 】

S 3 5 2 2 では、実行環境 4 0 0 は、U I 定義情報 3 4 1 1 をアプリユーザー用端末 2 0 0 に送信することなく、クライアント用 U I 定義情報 3 6 1 5 を実行環境 4 0 0 に送信する。これによって、図 3 6 に図示した通り、実行環境 4 0 0 に記録されたクライアント用 U I 定義情報 3 6 2 5 がアプリユーザー用端末 2 0 0 にダウンロードされ、クライアント用 U I 定義情報 3 6 3 5 として記録される。クライアント用 U I 定義情報 3 6 1 5 とクライアント用 U I 定義情報 3 6 3 5 とは同じ情報である。

【 0 4 6 0 】

S 3 5 2 3 ~ S 3 5 3 0 の処理は、前述した図 3 3 (b) の S 3 3 1 3 ~ S 3 3 2 0 とそれぞれ同様の処理であるため説明を省略する。

30

【 0 4 6 1 】

図 3 5 (c) の S 3 5 4 1 ~ S 3 5 4 3 は、前述した図 3 3 (c) の S 3 3 3 1 ~ S 3 3 3 3 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、S 3 5 4 3 では、クライアント用 U I 定義情報の取得要求を送信する。

【 0 4 6 2 】

S 3 5 4 4 は、アプリユーザー用端末 2 0 0 は、実行環境 4 0 0 からクライアント用 U I 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 U I 定義情報は前述した図 3 5 (b) の S 3 5 2 2 で実行環境 4 0 0 から送信されるものである。クライアント用 U I 定義情報を受信した場合は、クライアント用 U I 定義情報をメモリ 1 0 2 に記録して S 3 5 4 5 に進み、そうでない場合には S 3 5 4 4 でクライアント用 U I 定義情報の受信を待つ。クライアント用 U I 定義情報はメモリ 1 0 2（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 U R L 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 2 0 0 は、メモリ 1 0 2 に記録したクライアント用 U I 定義情報 3 6 3 5 に基づいてアプリの画面をディスプレイ 1 0 5 に表示する。

40

【 0 4 6 3 】

S 3 5 4 5 では、アプリユーザー用端末 2 0 0 は、前述した図 3 3 (c) の S 3 3 3 5 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

50

った場合には S 3 5 4 6 へ進み、そうでない場合には S 3 5 5 3 へ進む。

【 0 4 6 4 】

S 3 5 4 6 ~ S 3 5 5 3 の処理は、前述した図 3 3 (c) の S 3 3 4 3 ~ S 3 3 5 0 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 3 3 (c) の S 3 3 3 6 ~ S 3 3 4 2 に相当する処理は、変形例においてアプリケーション用端末 2 0 0 では行わない。

【 0 4 6 5 】

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

10

【 0 4 6 6 】

（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 C P U や M P U 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。

20

【 0 4 6 7 】

< 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。

【 0 4 6 8 】

[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、

30

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。

【 0 4 6 9 】

[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、

40

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。

【 0 4 7 0 】

[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、

50

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。

【 0 4 7 1 】

[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

10

を有することを特徴とする情報処理システム。

【 0 4 7 2 】

[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 のコンポー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

20

を有することを特徴とする情報処理システム。

【 0 4 7 3 】

[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

30

を有することを特徴とする情報処理システム。

【 0 4 7 4 】

[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）

40

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。

【 0 4 7 5 】

[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

50

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。

【 0 4 7 6 】

10

[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。

【 0 4 7 7 】

20

[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。

【 0 4 7 8 】

30

[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。

【 0 4 7 9 】

[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、C R U D ボタン配置によるボタンの配置、UI 画面の生成、等）、

40

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、

50

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。

【 0 4 8 0 】

[構成 1 1]

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、

10

を有することを特徴とする情報処理システム。

【 0 4 8 1 】

[構成 1 2]

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における前記雛形コンポー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、

20

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。

【 0 4 8 2 】

[構成 1 3]

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いない場合には、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

30

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。

【 0 4 8 3 】

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。

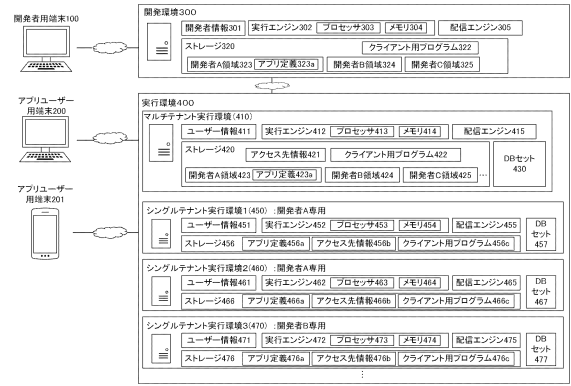
【 符号の説明 】

【 0 4 8 4 】

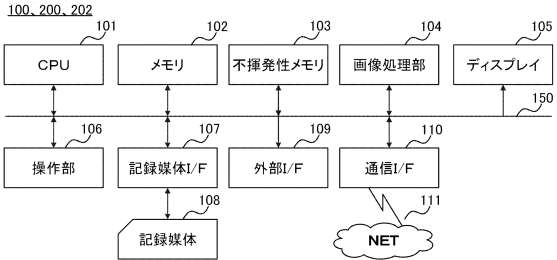
40

1 0 0 : 開発者用端末、 2 0 0 : アプリユーザー用端末、 2 0 1 : アプリユーザー用端末
2 0 1、 3 0 0 : 開発環境、 4 0 0 : 実行環境

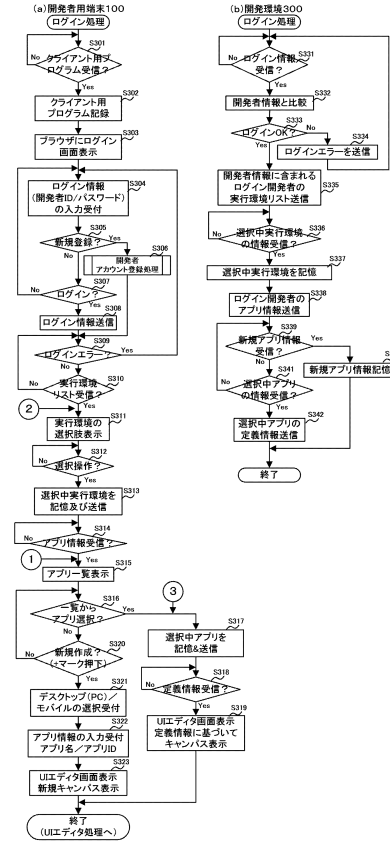
【図面】
【図 1】



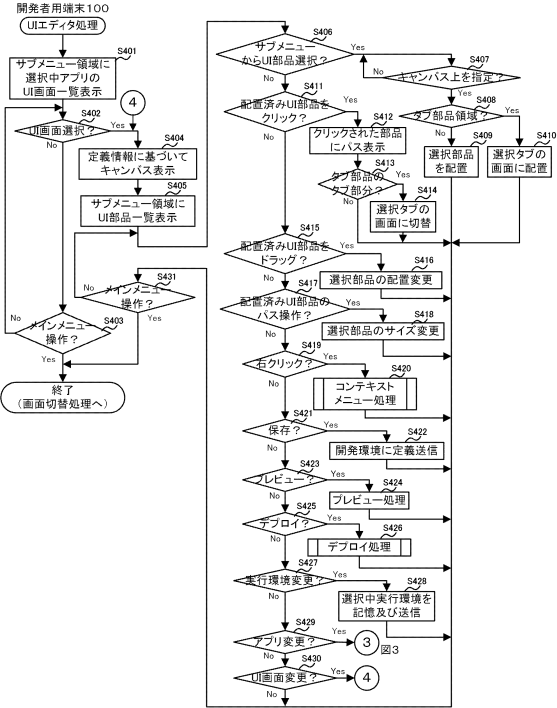
【図 2】



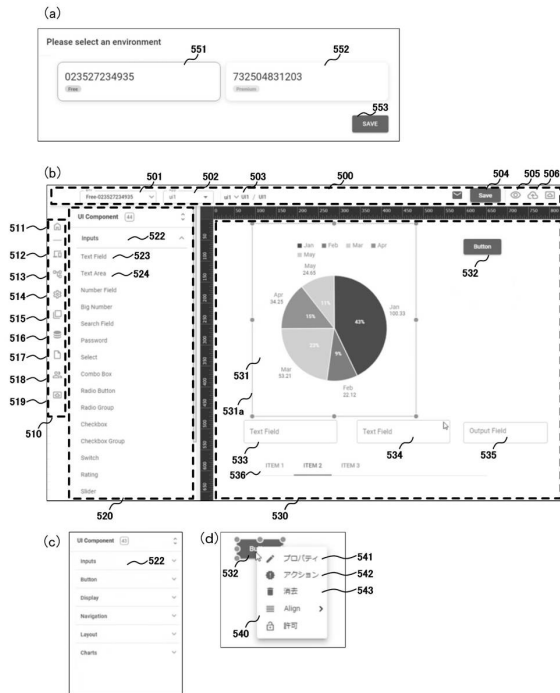
【図 3】



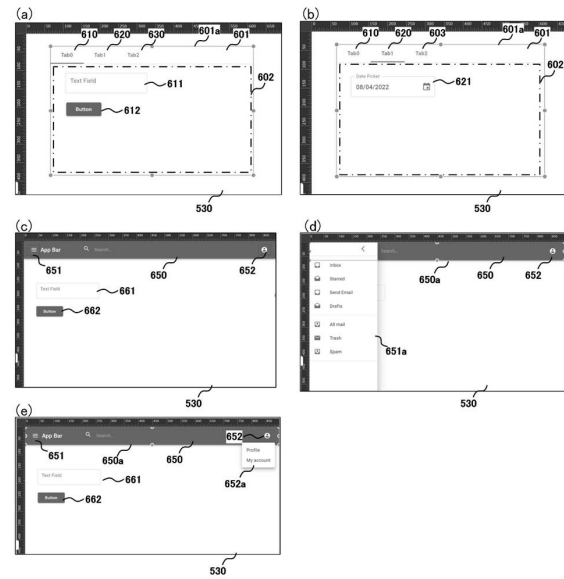
【図 4】



【図 5】



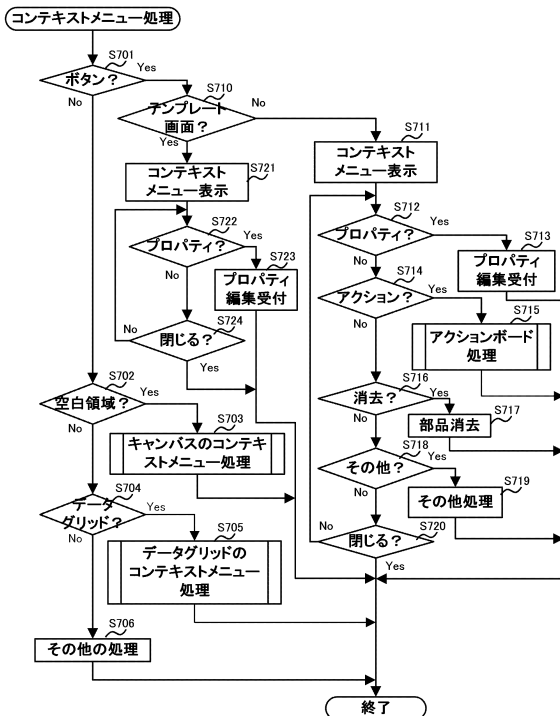
【図 6】



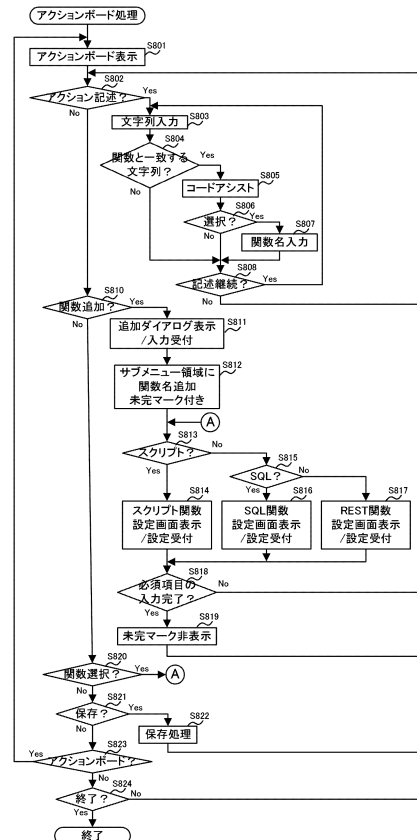
10

20

【図 7】



【図 8】

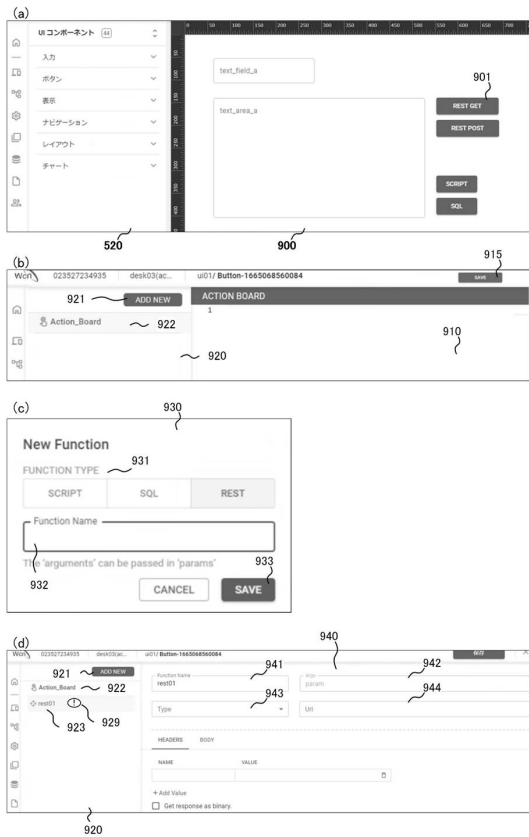


30

40

50

【 図 9 】



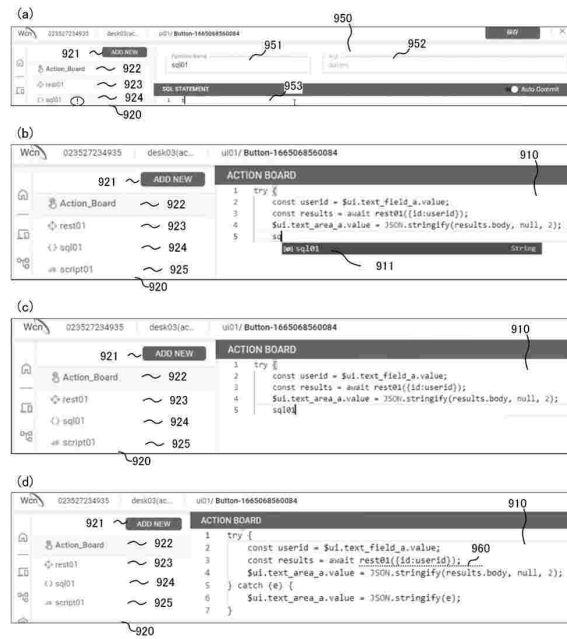
【 図 1 1 】

```

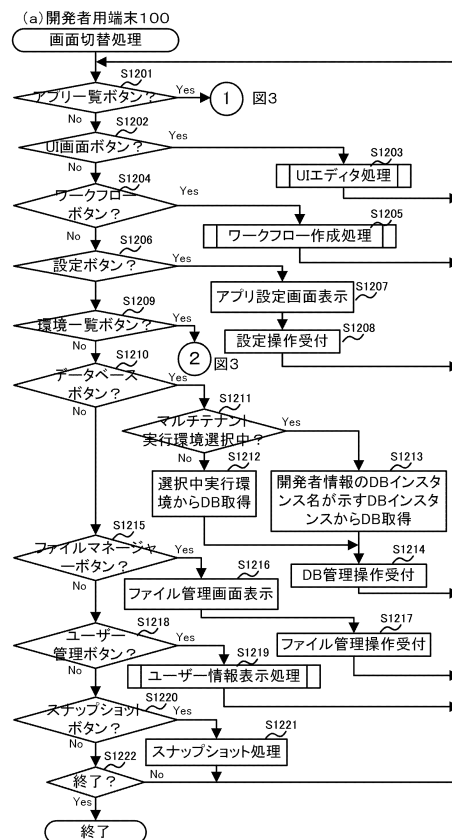
1  const res01 = async (param) => {
2    const toRestFnResult = (res) => {
3      return {
4        status: {
5          code: res.status,
6          text: res.statusText,
7        },
8        body: res.data,
9        headers: res.headers,
10      };
11    };
12
13    /** Parameters specified by dev user. */
14
15    const defaultHeaders = {
16      'Content-Type': 'application/json; charset=UTF-8',
17      'Access-Control-Allow-Origin': '*',
18    };
19
20    let inputData;
21    const contentType = 'none';
22    if (contentType === 'application/json' ||
23        contentType === 'text/html') {
24      contentType === 'text/css' ||
25      contentType === 'text/plain' ||
26      contentType === 'application/xml') {
27        inputData =
28          '';
29      }
30    } else {
31      let data = '{}';
32      inputData = JSON.parse(data);
33    }
34
35    const method = 'GET';
36    const url = 'https://jsonplaceholder.typicode.com/users/${param.id}';
37    const binaryResponse = false;
38
39    const headerParams = '{}';
40
41    const inputHeaders = JSON.parse(headerParams);
42
43    let headers = { ...defaultHeaders, ...inputHeaders };
44
45    let lambdaPayload = { method, url, data: inputData, headers: headers };
46    if (method === 'GET') {
47      delete lambdaPayload.data;
48    }
49    logger.log('CL>> lambdaPayload', lambdaPayload);
50
51    if (binaryResponse) {
52      lambdaPayload.responseType = 'arraybuffer';
53    }
54
55    const client = new LambdaClient();
56    const params = {
57      FunctionName: 'wpcn-lambda-rest',
58      InvocationType: 'RequestResponse',
59      LogType: 'Tail',
60      Payload: JSON.stringify(lambdaPayload),
61    };
62
63    const command = new InvokeCommand(params);
64    let resultToString;
65    try {
66      let responseData = await client.send(command);
67      resultToString = JSON.parse(Buffer.from(responseData.Payload));
68      logger.log('CL>> result', resultToString);
69    } catch (error) {
70      logger.error('CL>> REST function failed: ', error);
71      throw error;
72    }
73    return toRestFnResult(resultToString);
74  };

```

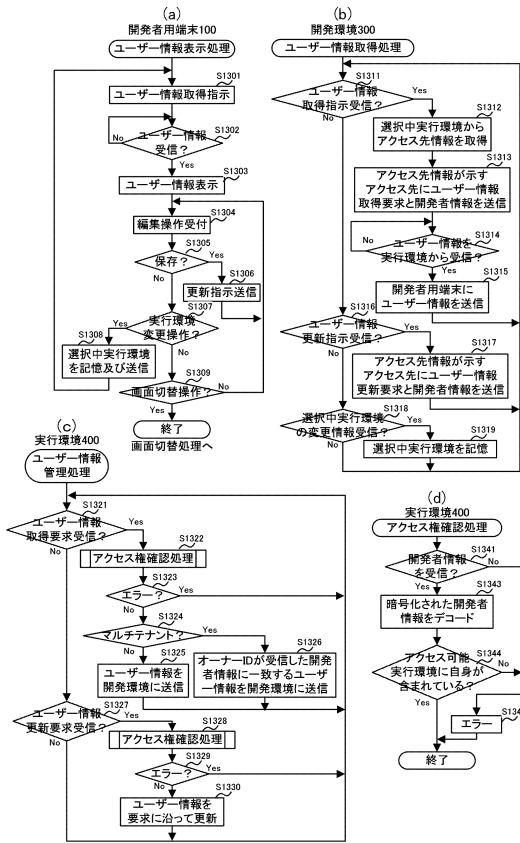
【 図 1 0 】



【 図 1 2 】



【図 1 3】



【図 1 5】

501 1501 1502 1503 1504

Username	Enabled	Account Status	Email Verified	Updated	Created	Delete
aaaa@xxx.co.jp	true	CONFIRMED	true	2022/10/7 8:19	2021/10/19 10:01	X
bbbb@xxx.co.jp	true	FORSE_CHANGE_PASSWORD	true	2021/10/20 11:02	2021/10/20 11:02	X
cccc@xxx.co.jp	true	CONFIRMED	false	2022/10/18 11:54	2021/10/21 13:05	X

518

【図 1 4】

(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 (1 2桁), シングルテナント実行環境 1 のID (1 2桁), シングルテナント実行環境 2 のID (1 2桁)	confirmed	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 (1 2桁), シングルテナント実行環境 3 のID (1 2桁)	confirmed	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)

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
...

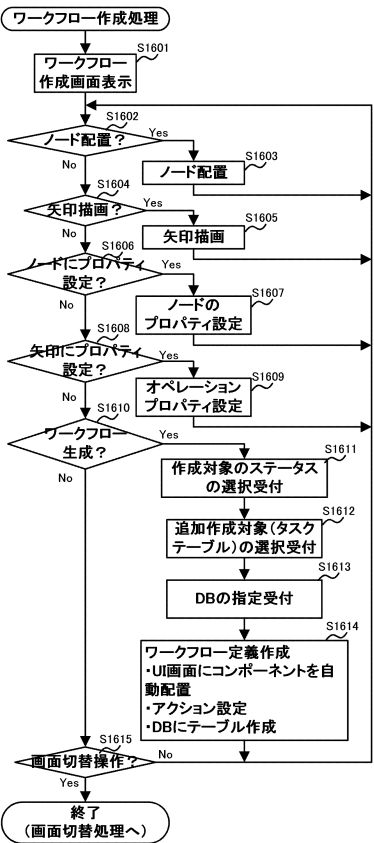
(c) ユーザー情報 (シングルテナント実行環境に保持)

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
Agf	aaaa@xxx.co.jp; bbbb@xxx.co.jp
Bgr	cccc@xxx.co.jp
Cgr	aaaa@xxx.co.jp; cccc@xxx.co.jp
...	...

【図 1 6】



10

20

30

40

50

【図 17】



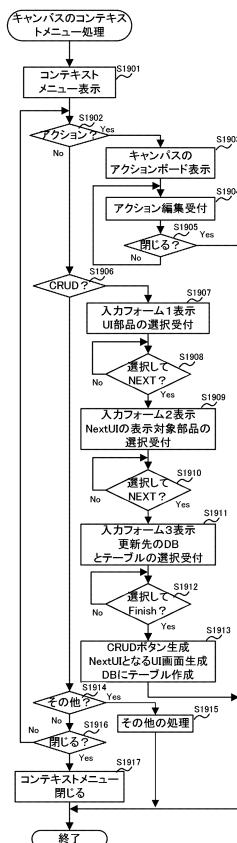
【図 18】



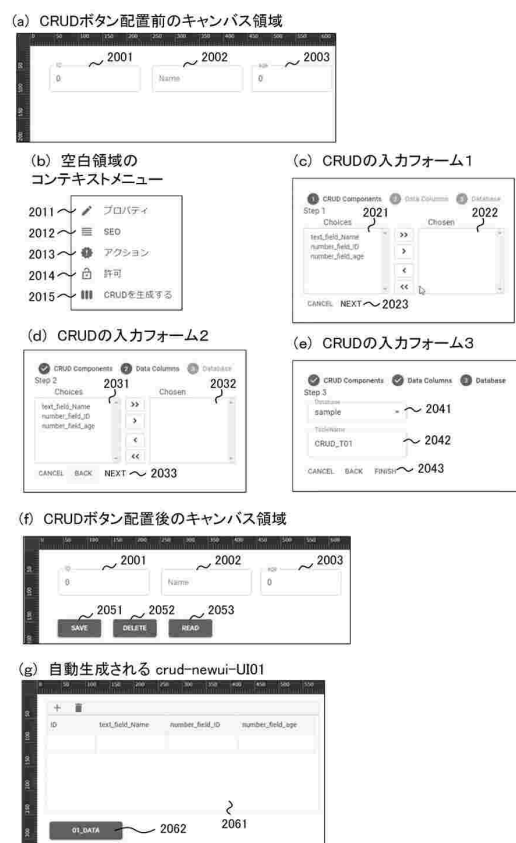
10

20

【図 19】



【図 20】



30

40

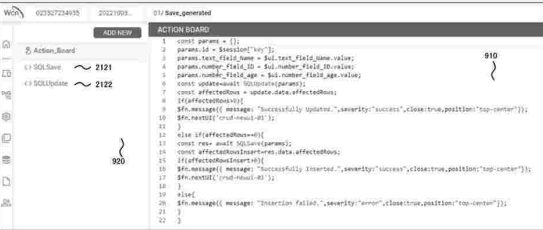
50

【 図 2 1 】

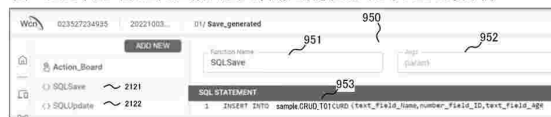
(a) CRUDボタン配置によって追加されるDBテーブルの例



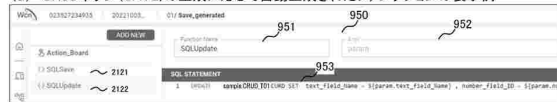
(b) CRUDボタン(SAVE)のアクションボードの表示例



(c) CRUDボタン(SAVE)の生成に応じて自動生成された関クションの表示例

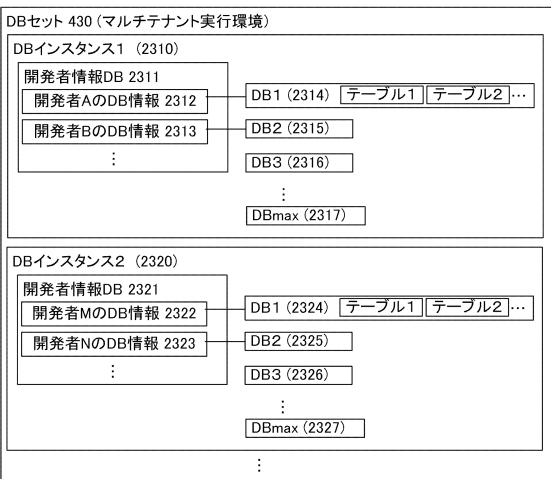


(d) CRUDボタン(SAVE)の生成に応じて自動生成された関クションの表示例

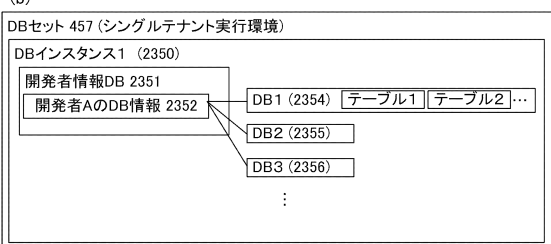


【 図 2 3 】

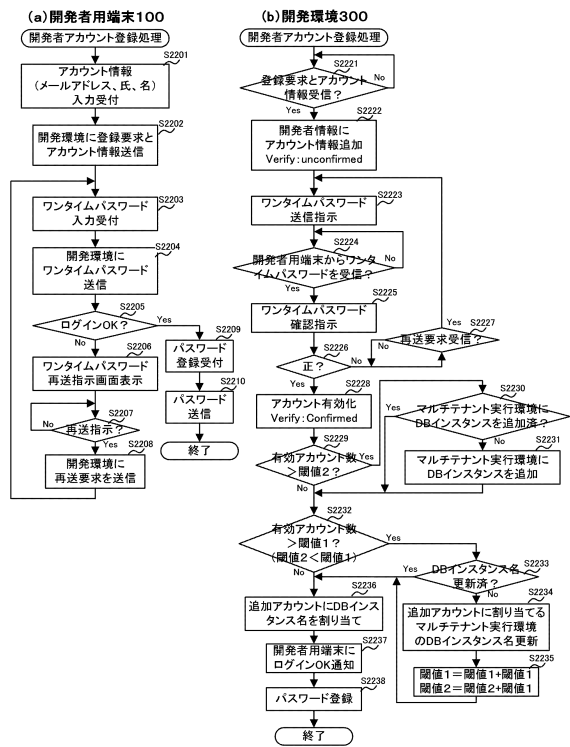
(a)



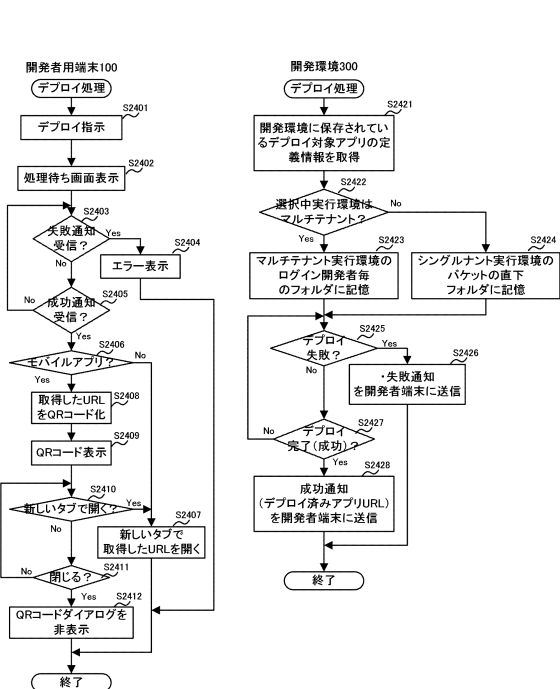
(b)



【 图 2 2 】



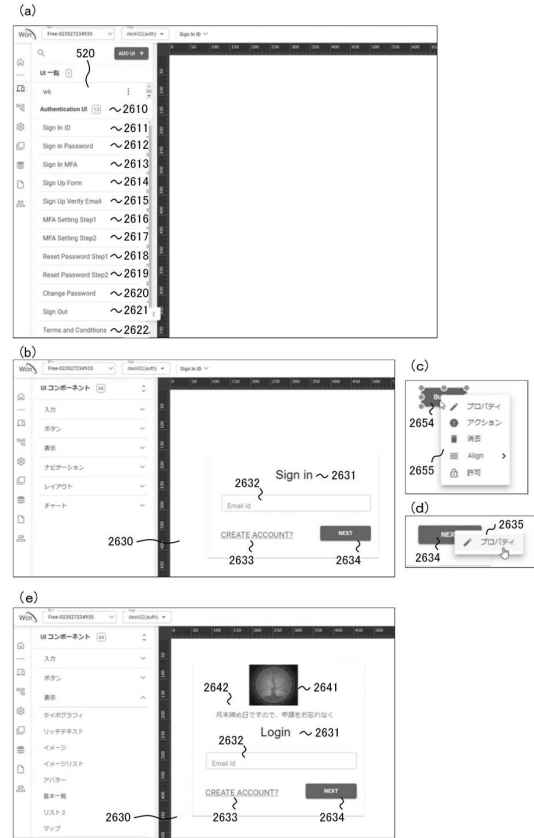
【 図 2 4 】



【図 25】



【図 26】



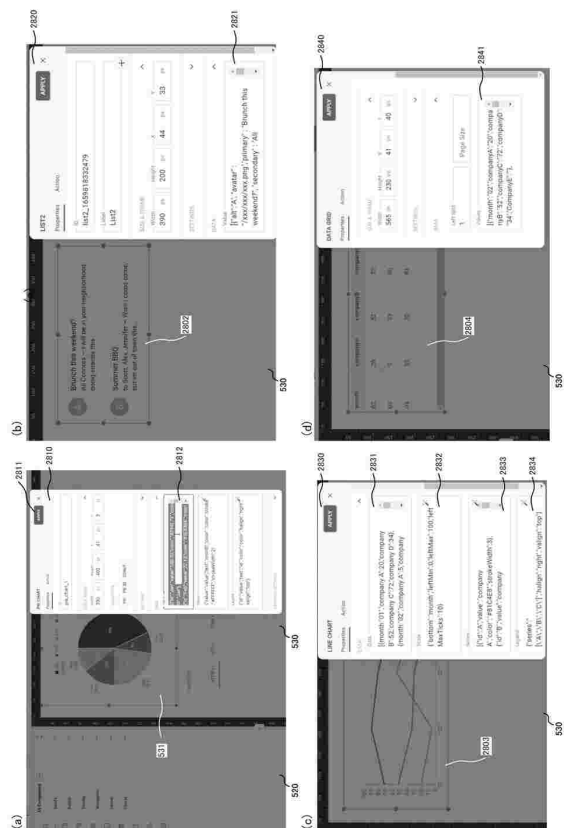
10

20

【図 27】



【図 28】

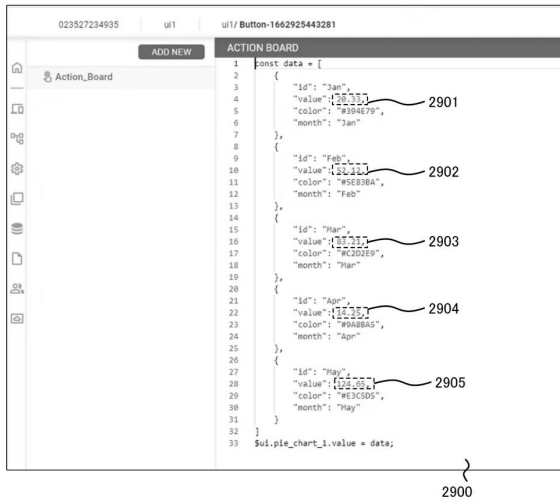


30

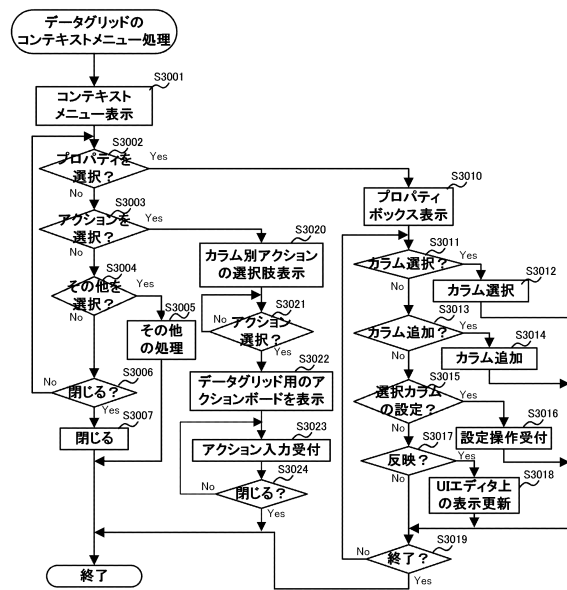
40

50

【図 29】



【図 30】



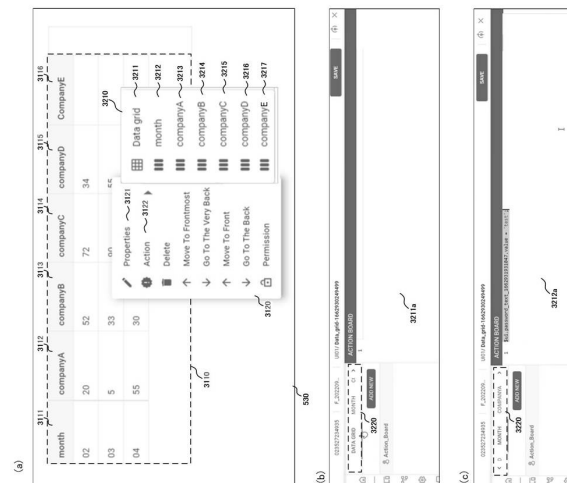
10

20

【図 31】



【図 32】

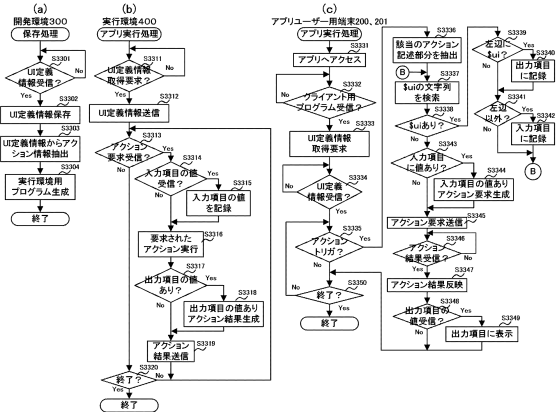


30

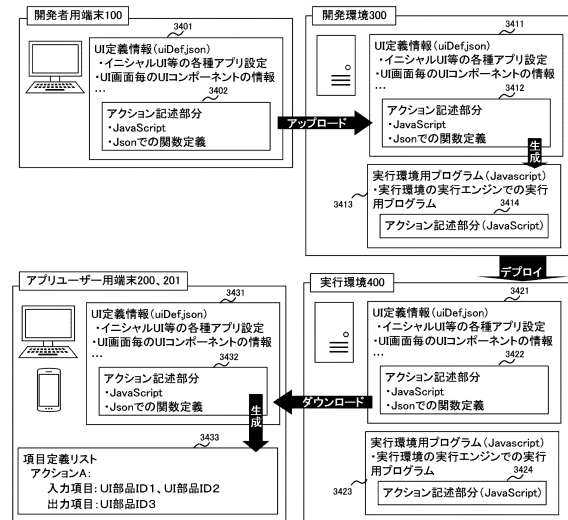
40

50

【図 3 3】



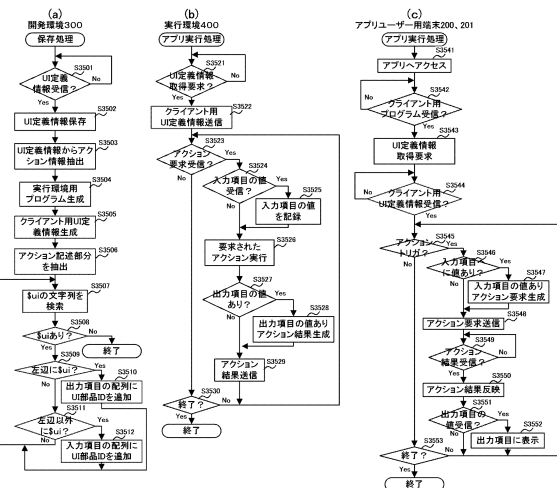
【図 3 4】



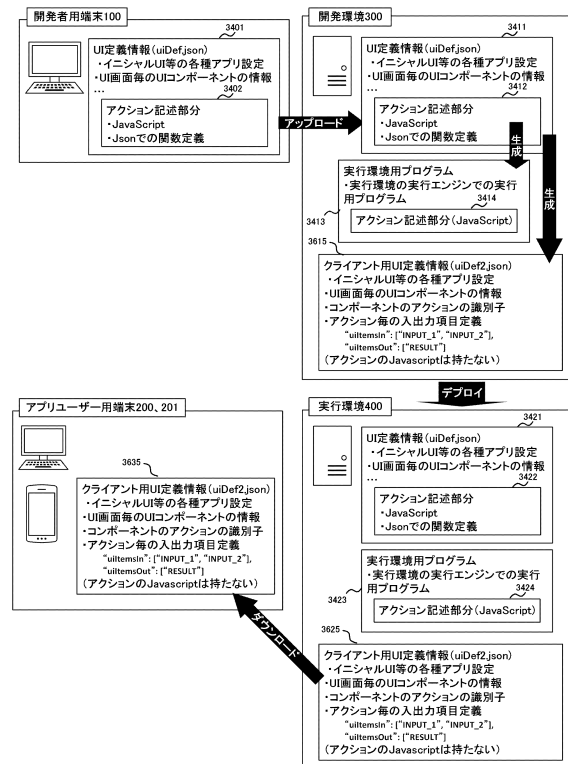
10

20

【図 3 5】



【図 3 6】



30

40

50

フロントページの続き

東京都港区港南 2 丁目 1 6 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 鈴木 智子
東京都港区港南 2 丁目 1 6 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 BC51