配達支援システム
内部設計書

第 0 版

ONO-Systems

平成 30 年 12 月 7 日

目 次

1　開発対象のシステム概要

本システムは，配達物の配達支援を行うシステムです．主な機能を以下に示します．

- 消費者への通知機能
- 配達員の位置情報の表示機能
- 消費者の受取選択結果のリアルタイム表示機能

2　開発環境

本システムの開発環境を以下に示します．

- Android アプリケーション
  Android Studio ver4.2
- サーバ
  AWS(AmazonWebService)
- 開発言語
  Java，MySQL
- 文書・コード管理
  GitHub

3　動作環境

- Android アプリケーション
  Android 4.2
- サーバ
  AWS(AmazonWebService)

4　コーディング規約

本プロジェクトのプログラムは，以下の規則を遵守します．

4.1　コンポーネントやクラスについて

- UpperCamelCase を利用する
- Component 名は最後につける (Activity，Fragment，TextArea など)
- 本プロジェクトで頻繁に利用する名詞は以下の表を用いて命名する (表 1 参照)

表 1: 本プロジェクトで利用する名詞の英単語表

内部設計書での名前	英単語
アカウント保持者	User
消費者	Customer
配達員	Courier
配達物	Delivery

4.2 変数の命名について

- 命名には英語を用いる
- 名前から役割が読み取れるように命名する
- グローバル変数に用いる英単語は省略しない (String ⇒ Str など)
- LowerCamelCase を利用する
- 定数は全て大文字で表し，スネークケースで連結する
- for や while のカウンタ変数には，i，j，k を用いる

4.3 関数の命名について

- 意味と英単語の対応付けを統一する (表 2)
- 名前は動詞で始める
- LowerCamelCase を利用する

表 2: 命名に使う英単語表

意味	英単語	例
真偽値を取得	is	#isCreated
値を代入	set	#setDelivery
引数を含むか判定	has	#hasFragment
特定の動作をしたときに動作	on	#onClick
新しく作る	create	#createSubActivity
新しく作る	new	#newDelivery
更新	update	#updateDeliveryDate
サーバから情報を取得	fetch	#fetchDeliveryList

4.4 コーディングについて

- 字下げは半角スペース 2 つを用いる
- マジックナンバーは使用しない

- 安易にネット上のソースコードを利用しない
- class や if，while などのブロック始点のブラケット ('{') は改行せずに，半角スペースを空けて記述する
- 機能が完成したとき (Pull Requests を出すとき) にデバッグ用のコードを残さない

4.5　コメントについて

- 複雑な処理はコメントをつける
- メソッドにはコメントをつける
- TODO:など，コードの説明でないコメントは PullRequests を出すときには消す

5　ネットワーク設計

本システムのネットワークは図 1 のように構成されます.

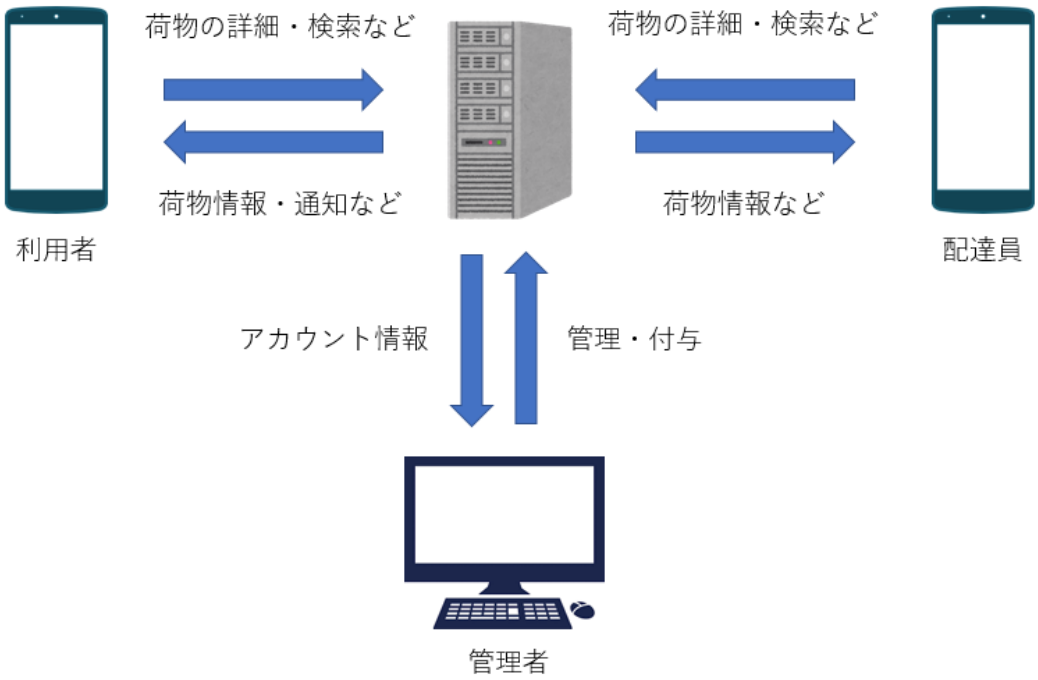

図 1: ネットワーク構成図

6　Android モジュール設計

Android のモジュール設計を以下に示します.

6.1　モジュール構成

本システムでの画面遷移は，図 2，3 のようになっています．各画面を構成するモジュールについて説明します．また，サーバとの通信に必要なモジュール，通知機能に関するモジュールなども設計します．画面遷移する際は遷移先のモジュール呼び出しを行います．本システムは消費者側と配達員側で遷移する画面が異なりますが，類似する機能が多数存在します．そのため，共通部分は１つのモジュールとして作成し，異なる機能の実装は元となるモジュールを継承し消費者側と配達員側で分けて作成しています．

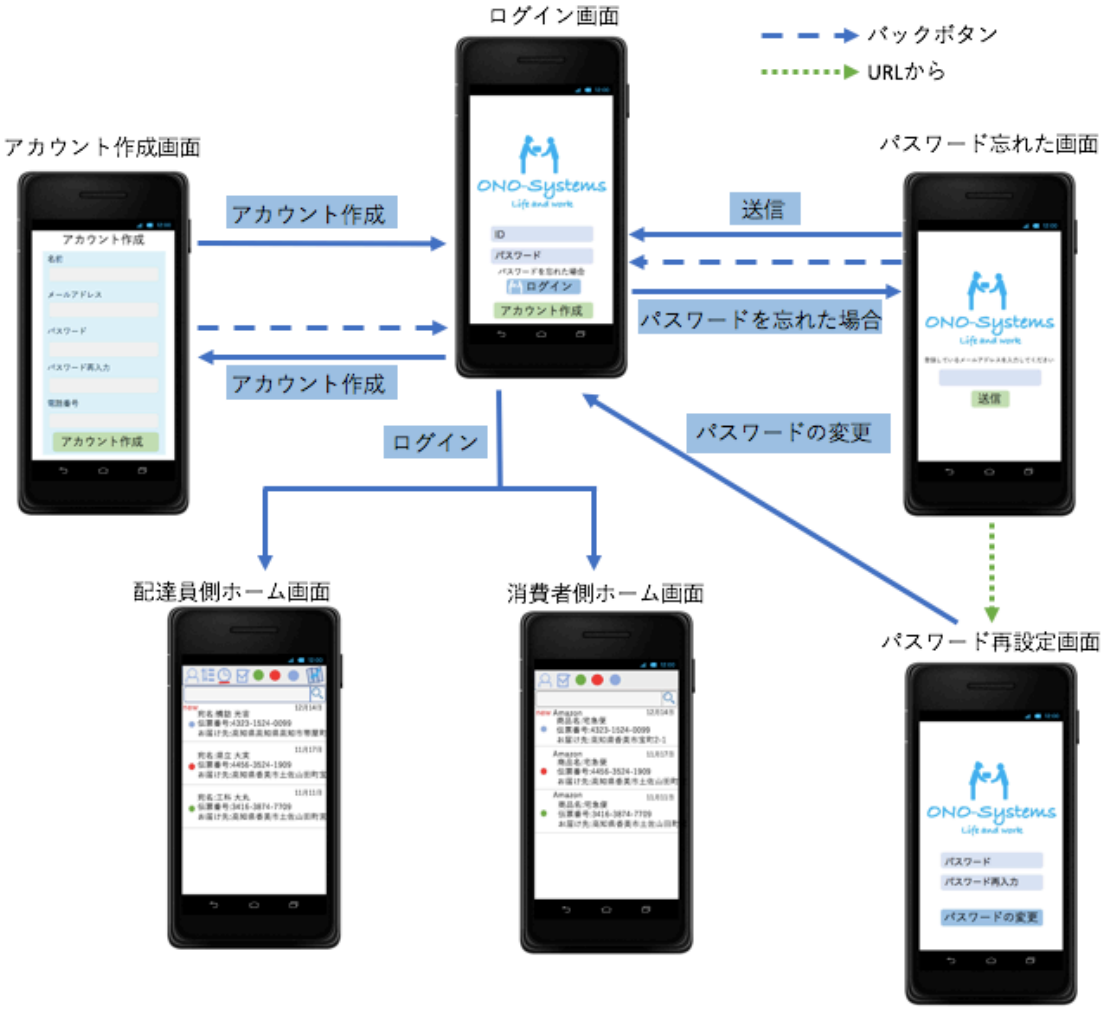

図 2: ログイン画面からの画面遷移図

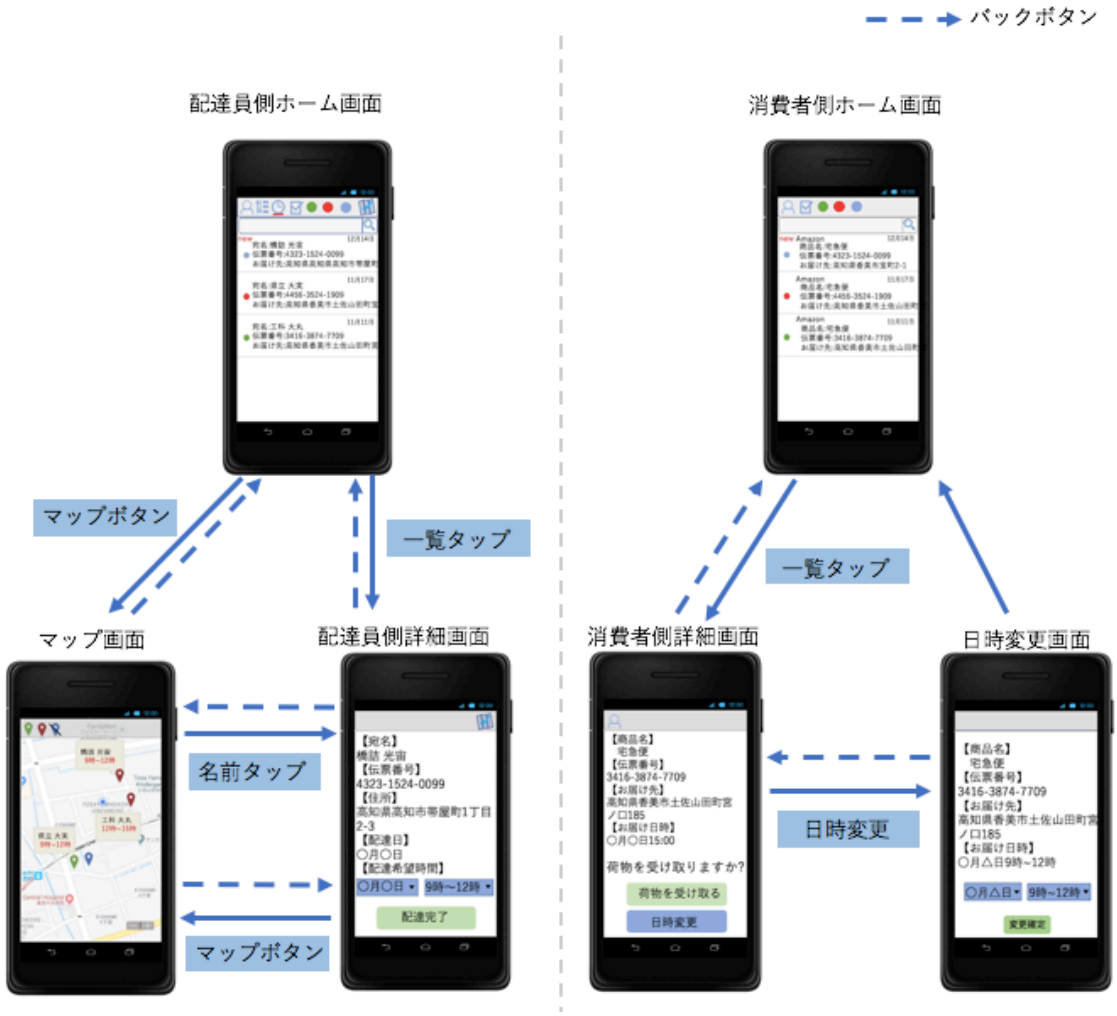

図 3: ホーム画面からの画面遷移図（左：配達員側，右：消費者側）

6.2 モジュール仕様

Android の各モジュールの仕様は以下の通りです.

7 Server モジュール設計

Server のモジュール設計を以下に示します．引数や返り値で使用している単語は，表 3 の通りです．receivable_status は，未配達を 0，配達済みを 1 で表します．deliveied_status は，未選択の状態を 0，受け取れない場合は 1，受け取れる場合は 2 で状態を表現します.

表 3: 本節で使用する変数の定義

型	変数名	意味
int	driver_id	配達員の ID
int	costomer_id	消費者の ID
string	name	名前
string	possword	パスワード
string	mail	メールアドレス
string	tel	電話番号
string	address	住所
long	slip_number	伝票番号
long	ship_from	発送元
int	time	配達日・時間
int	receivable_status	配達状況
int	deliveied_status	受け取り可否

7.1 モジュール構成

サーバは，認証，配達員側，消費者側のモジュールに加え，管理者側や通知機能に関するモジュールから構成されています.

7.2 モジュール仕様

Server の各モジュールの仕様は以下の通りです.

7.2.1 認証機能

ログインを行う際に使用する認証系のモジュールについて示します．ログイン機能では，配達員側，消費者側どちらも同じモジュールを用いて認証を行います.

- Login.java
  動作： 初めにハッシュ値を計算しているか確認を行う．計算していなければパスワードからハッシュ値の計算を行いデータベースに値を格納する．次に，ID とパスワードからデータベースにあるハッシュ値を比較し，ログイン処理を行う．最後に，セッション情報をサーバ内に記憶させる.
  引数： id，password
  返り値： customer_id，または，driver_id，プラス user_type (NULL 値なら認証不可)

- AuthFilter.java
  動作： 認証が通っているかの判別を各サーブレットについて行う．判別にはセッション情報を用いる．このプログラムは，認証が必要な全てのサーブレットに対して行う.
  引数： なし
  返り値： なし

- RegisterAccount.java
  動作： 入力された情報からデータベースに新たな消費者の登録を行う．また，パスワードからハッシュ値を計算しデータベース内に格納する．
  引数： name，mail，password，tel，address
  返り値： なし

7.2.2 配達員側

配達員側のモジュールを以下に示します．

- TopCourier.java
  動作： 当日に配達する荷物情報を，データベースからドライバー ID を参照して送信する．
  引数： driver_id
  返り値: name, slip_number, ship_from, address, time, receivable_status, deliveied_status

- SettingCourier.java
  動作： 配達員が入力した情報からデータベース内の配達員情報の変更を行う．
  引数： (配達員の)name，mail，password，tel
  返り値： なし

- ChangeTimeCourier.java
  動作： 入力された情報からデータベース内の配達希望日時の変更を行う．
  引数： slip_number，time
  返り値： time

- CompleteCourier.java
  動作： 伝票番号からデータベース内の荷物情報に配達済みのフラグを立てる．
  引数： slip_number
  返り値： なし

- InformationCourier.java
  動作： データベースから配達員の情報を取得し，送信する．
  引数： driver_id
  返り値： name，mail，tel

7.2.3 消費者側

消費者側のモジュールを以下に示します．

- TopCustomer.java
  動作： 入力された情報から利用者の荷物情報を送信する．
  引数： customer_id
  返り値: name, slip_number, ship_from, address, time, receivable_status, deliveied_status

- SettingCustomer.java
  動作： 入力された情報からデータベース内の消費者情報の変更を行う．

引数： (消費者の)name，mail，password，tel，address
返り値： なし

- ReceiveCustomer.java
  動作： 伝票番号から荷物受け取りの可否をデータベース内に格納する．
  引数： slip_number
  返り値： なし

- ChangeTimeCustomer.java
  動作： 消費者が入力した情報からデータベース内の配達希望日時の変更を行う．
  引数： slip_number，time
  返り値： time

- InformationCustomer.java
  動作： データベースから利用者の情報を取得し，送信する．
  引数： customer_id
  返り値： name，mail，tel，address

7.2.4 通知機能

通知機能のモジュールを以下に示します．

- Notification.java
  動作： トークンを受け取り，通知を送ってもらうよう Firebase に対してデータを送信する．
  引数： token
  返り値： token，text

7.2.5 管理者側

管理者側のモジュールを以下に示します．

- LoginManager.jsp
  動作： 管理者ページへのログインを行う．
  引数： id，password
  返り値： なし

8 データベース設計

本システムではデータベースに AWS(AmazonWebService) を使用します．
ER 図など

8.1 各テーブルの詳細

本システムのデータベースには，5 個のデータテーブルを用います．各データテーブルの役割と属性を以下に示します．

8.1.1 消費者テーブル

消費者テーブルでは，消費者に関する情報を管理します．このテーブルのデータテーブルを表 4 に示します．

表 4: 消費者テーブル

属性	データ型/長	NULL	Key	初期値	その他
customer_id	int(9) unsigned	NO	PRIMARY	NULL	auto_increment
name	varchar(64)	NO			
mail	varchar(64)	NO			
tel	varchar(11)	YES		NULL	
address	varchar(128)	YES		NULL	
hash	varchar(255)	NO			
salt	varchar(255)	NO			

8.1.2 配達員テーブル

配達員テーブルでは，配達員に関する情報を管理します．このテーブルのデータテーブルを表 5 に示します．

表 5: 配達員テーブル

属性	データ型/長	NULL	Key	初期値	その他
driver_id	int(9) unsigned	NO	PRIMARY	NULL	auto_increment
name	varchar(64)	NO			
mail	varchar(64)	NO			
tel	varchar(11)	YES		NULL	
store_code	varchar(64)	YES		NULL	
account_type	int(1) unsigned	YES		NULL	
hash	varchar(255)	NO			
salt	varchar(255)	NO			

8.1.3 商品テーブル

商品テーブルでは，商品に関する情報を管理します．このテーブルのデータテーブルを表 6 に示します．

表 6: 商品テーブル

属性	データ型/長	NULL	Key	初期値	その他
slip_number	varchar(12)	NO	PRIMARY	NULL	
name	varchar(64)	NO			
address	varchar(128)	NO			
time	varchar(11)	YES		NULL	
delivery_status	int(1) unsigned	NO		0	
receive_status	int(1) unsigned	NO		0	
customer_id	int(9) unsigned	NO	MULTIPLE	NULL	
driver_id	int(9) unsigned	NO	MULTIPLE	NULL	

8.1.4 管理者テーブル

管理者テーブルでは，管理者に関する情報を管理します．このテーブルのデータテーブルを表 7 に示します．

表 7: 管理者テーブル

属性	データ型/長	NULL	Key	初期値	その他
manager_id	int(1) unsigned	NO	PRIMARY	NULL	auto_increment
name	varchar(64)	NO			
mail	varchar(64)	NO			
store_code	varchar(64)	YES		NULL	
account_type	int(1) unsigned	YES		NULL	
hash	varchar(255)	NO			
salt	varchar(255)	NO			

8.1.5 地図テーブル

地図テーブルでは，地図に関する情報を管理します．このテーブルのデータテーブルを表 8 に示します．

表 8: 地図テーブル

属性	データ型/長	NULL	Key	初期値	その他
map_id	varchar(12)	NO	MULTIPLE	NULL	
lat	double(8,6)	YES		NULL	
lng	double(9,6)	YES		NULL	

9　バージョン管理規約

本システムの開発では，GitHub を用いてファイルの管理を行います．GitHub を使用する際には，以下の規則を遵守します.

- ドキュメント関連の資料は，onosystem-doc で管理する
- Android のソースコードは，onosystem-android で管理する
- Server のソースコードは，onosystem-server で管理する
- 編集作業を行う際には，ブランチを切ってコミットする
- 開発用ブランチの名前は,「dev_○○（開発している機能名）」にする
- 細かい頻度でコミットする（1 日の作業ごとに纏めてコミットしない）
- コミットのコメントはわかりやすい内容にする
- Pull Requests されたものを確認し，評価をリアクションのアイコンを追加することで示す
- Pull Requests に対して高評価が 3 つ以上ある場合には master に Merge する