配達支援システム
内部設計書

第 2.0 版

ONO-Systems

平成 30 年 12 月 16 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
  配達物が届く際に，消費者の端末に配達物に関する通知を送ります．消費者はその際に，配達物を受け取れるかどうかを選択することができます．受け取ることができない場合は，配達日時を変更することが可能です．

- 消費者の受取選択結果のリアルタイム表示機能
  消費者が選択した受取可否の情報を，配達員側の端末にリアルタイムに表示します．受け取る場合，受け取らない場合，未選択の場合を異なる色で表現し，受取選択結果の一覧を表示します．

- 配達先の位置情報の表示機能
  配達員側の機能に，マップ画面で配達先の位置を表示する機能があります．受取可否ステータス別に表示でき，消費者が受け取らない選択をした配達物を非表示にする機能を実現します．

- 配達物のソート機能
  配達員側では，配達物一覧を，配達先の距離順に並びかえる機能があります．また，配達物の指定時間順でソートする機能も実装します．

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 ver3.0 以上

- サーバ
  AWS(AmazonWebService)

- 開発言語
  Java，MySQL

- 文書・コード管理
  GitHub

- サービス
  Firebase，Google Maps API

3　動作環境

- Android アプリケーション
  API レベル 24 以上

- 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
- データベースと対応する変数名は，スネークケースを利用する
- 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.

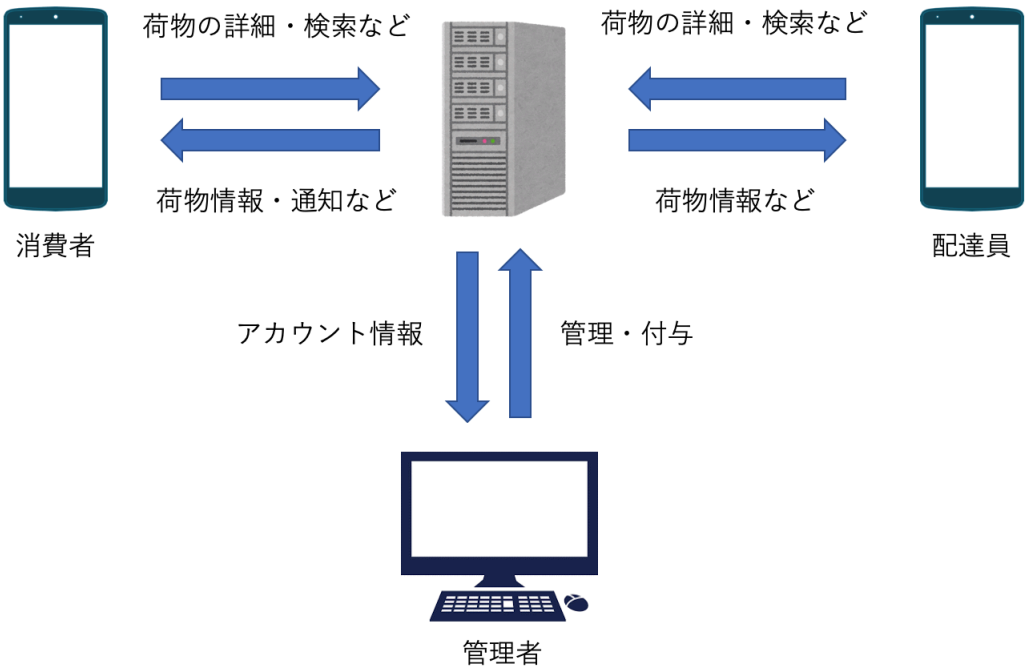

図 1: ネットワーク構成図

6 Android モジュール設計

Android のモジュール設計について示します.

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，共通部分は一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.

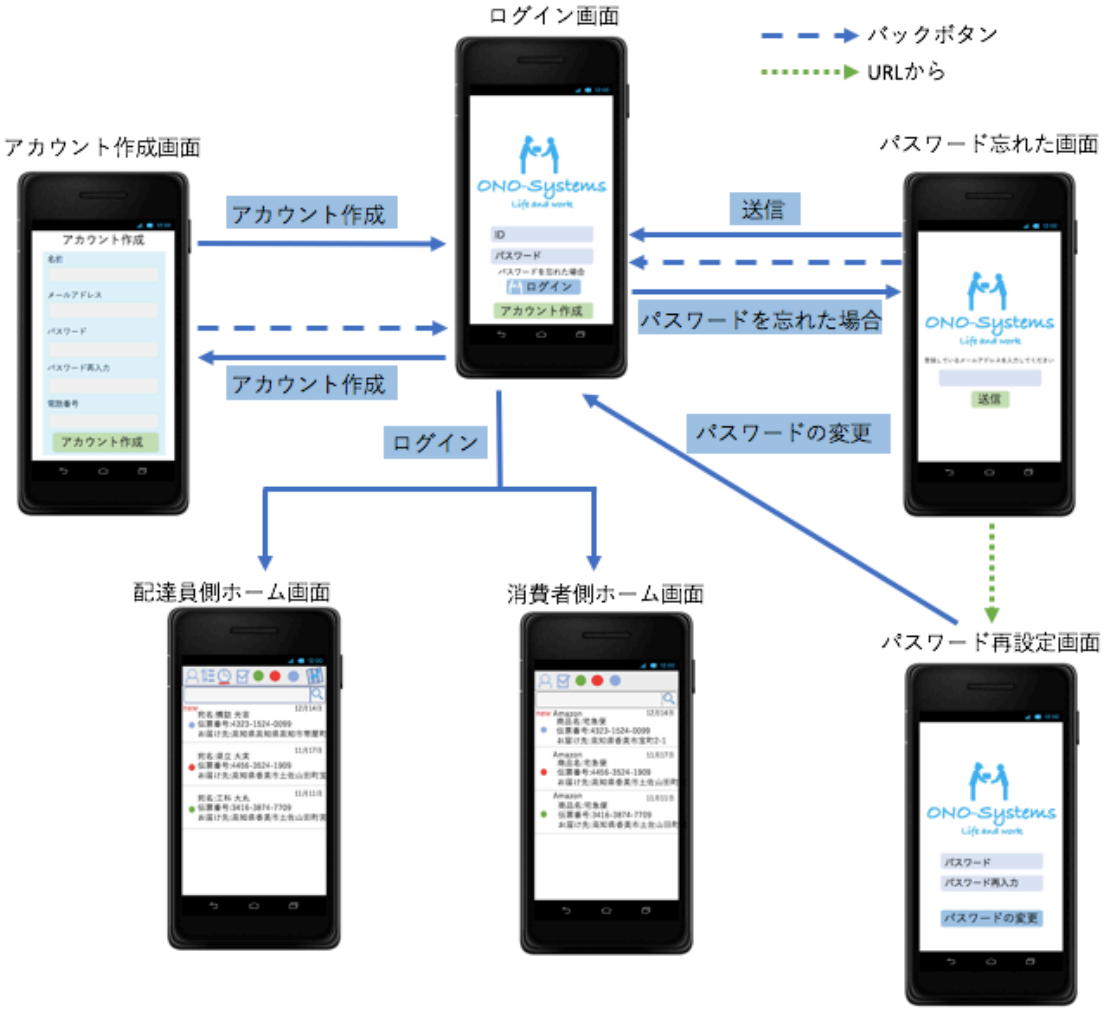

図 2: ログイン画面からの画面遷移図

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について以下に示します．

モジュール仕様で示す各メソッドでは，動作，引数，返り値について説明しています．なお，メソッドの引数に，View view と記載している部分がありますが，この view とは，画面上に配置しているボタンのことです．Android Studio では，ボタンが押された場合の処理を行う際，引数に View 型が渡される仕様であるため，本小節で示す複数の Button メソッドでは，引数に View view と記載しています．

6.2.1 Delivery クラス

配達物に関する情報を持つクラスです．Delivery クラスで定義する要素を，表 3 に示します．

表 3: Delivery クラス

型	変数・定数名	意味
long	slip_number	伝票番号
string	name	宛名
string	address	住所
long	ship_from	発送元
int	time	配達日時
int	receivable_status	受取可否
int	delivered_status	配達状況
double	lat	緯度
double	lng	経度
boolean	visible	true：表示，false：非表示
int	UNDELIVERED	0：未配達
int	DELIVERED	1：配達済み
int	UNSELECTED	0：未選択
int	UNRECEIVABLE	1：受け取り不可
int	RECEIVABLE	2：受け取り可

6.2.2 User クラス

ユーザに関する情報を持つクラスで，消費者側・配達員側で共通の要素を定義します．非共通部分はそれぞれのクラスで実装します．User クラスで定義する要素を，表 4 に示します．

表 4: User クラス

型	変数名	意味
string	name	名前
string	password	パスワード
string	mail	メールアドレス
long	tel	電話番号

6.2.3 Customer クラス

消費者に関する情報を持つクラスです．User クラスを継承して作成します．Customer クラスで定義する要素を，表 5 に示します．

表 5: Customer クラス

型	変数名	意味
int	customer_id	消費者 ID
string	address	住所

6.2.4 Courier クラス

配達員に関する情報を持つクラスです．User クラスを継承して作成します．Courier クラスで定義する要素を，表 6 に示します．

表 6: Courier クラス

型	変数名	意味
int	driver_id	配達員 ID
int	store_code	店舗コード
int	account_type	アカウントの種類

6.2.5 ResponseData クラス

Request クラスの sendRequest メソッドの返り値が，ResponseData クラスのオブジェクトです．ResponseData クラスで定義する要素を，表 7 に示します．

表 7: ResponseData クラスで定義する変数

型	変数名	意味
String	json	レスポンスのボディー
int	httpStatusCode	HTTP ステータスコード

6.2.6 Request クラス

サーバにリクエストを投げる際に使用するクラスです.

- setSessionId
  動作： セッション ID を設定する static メソッド.
  引数： int sessionId
  返り値： なし

- sendRequest
  動作： サーバにリクエストを投げる．body は json で渡す．セッション ID があれば付与してリクエストする．返り値は ResponseData クラスのオブジェクト.
  引数： String URL, String json
  返り値： ResponseData responseData

6.2.7 LoginActivity クラス

起動時に呼び出され，ログイン画面とログイン機能を実現するクラスです．ログイン完了後には，ホーム画面へと遷移します．LoginActivity クラスのメソッドは以下の通りです.

- loginButton
  動作： ログインボタンが押された際に，フォームに入力されたユーザ ID とパスワードを引数に渡して login を呼び出す.
  引数： View view
  返り値： なし

- autoLogin
  動作： 端末内に保持されているユーザ ID とパスワードを引数に渡して login を呼び出す.
  引数： なし
  返り値： なし

- login
  動作： サーバにユーザ ID とパスワードを送信し，ログインを試行する．ログインに成功した場合，transitionActivity を呼び出す.
  引数： int id(customer_id または driver_id)，String password
  返り値： なし

- sendToken
  動作： 通知を送信するための端末固有のトークンをサーバに送信する.
  引数： なし
  返り値： なし

- transitionActivity
  動作： ログインに成功した場合，ホーム画面に遷移する．
  引数： int sessionId
  返り値： なし

- createNewAccountActivity
  動作： ログイン画面のアカウント作成ボタンが押された際，アカウント作成画面へ遷移する．
  引数： なし
  返り値： なし

6.2.8 NewAccountActivity クラス

アカウント作成画面を定義するクラスで，新規アカウント作成機能を実現します．

- createNewAccountButton
  動作： アカウント作成画面のアカウント作成ボタンが押された際 createNewAccount を呼び出す．アカウント作成終了後，ログイン画面に戻る．
  引数： なし
  返り値： なし

- createNewAccount
  動作： 入力されたデータで新規アカウントの作成を行う．
  引数： String name，String mail，long tel，String address，String password
  返り値： なし

6.2.9 HomeActivity クラス

ホーム画面を定義する継承元のクラスで，消費者側・配達員側の共通機能を実装します．ホーム画面は，配達物の一覧を表示する画面です．配達物の受取可否のステータス別に表示する機能や検索機能，プロフィール情報変更機能を実現します．消費者側・配達員側の非共通機能はこのクラスを継承し，それぞれのクラスで実装します．HomeActivity クラスのメソッドは以下の通りです．

- reloadDeliveries
  動作： deliveriesInfo から配達物一覧を表示する．deliveriesInfo は getDeliveries メソッドから取得する．
  引数： なし
  返り値： なし

- getDeliveries
  配達物一覧の取得を行う abstract メソッド

- editProfile
  プロフィール情報の編集を行うための abstract メソッド

- updateProfile
  プロフィール情報を更新する abstract メソッド

- findDeliveryButton
  動作： 検索窓に入力された内容に該当する配達物を表示させるボタン．押されたら findDeliveries メソッドを呼び出す．また，refresh メソッドを呼び出し更新する．
  引数： View view
  返り値： なし

- findDeliveries
  動作： 検索窓に入力された情報と一致する deliveriesInfo(ship_from) の visible を true にする．
  引数： 検索窓に入力された情報
  返り値： なし

- refresh
  動作： ホーム画面を再描画する．指定した間隔で reloadDeliveries メソッドを呼び出す．
  引数： なし
  返り値： なし

- editProfileButton
  動作： Profile 編集画面を出すためのボタン．押されたら editProfile メソッドを呼び出す．
  引数： View view
  返り値： なし

- receivableSelectButton
  動作： 受取可否のステータス別に表示するボタン．view からどのボタンが押されたかを判定し，toggleVisibleFromReceivable メソッドを呼び出す．
  引数： View view
  返り値： なし

- toggleVisibleFromReceivable
  動作： 該当ステータスの表示非表示 (visible) を切り替え，refresh メソッドを呼び出し再描画する．
  引数： int delivered_status
  返り値： なし

- showDeliveryDetailActivity
  動作： 配達物詳細画面へ遷移する．
  引数： Delivery delivery
  返り値： なし

6.2.10 CustomerHomeActivity クラス

消費者側のホーム画面を定義するクラスで，HomeActivity クラスを継承して実装します．CustomerHomeActivity クラスのメソッドは以下の通りです．

- getDeliveries
  動作： 消費者 ID を用いてデータベースから配達物一覧の取得を行う.
  引数： int customer_id
  返り値： ArrayList<Delivery> deliveriesInfo

- editProfile
  動作： customer から引数を取得する．また，getProfileCustomer と updateProfile のメソッドを呼び出して変更を行う.
  引数： Customer customer
  返り値： なし

- getProfileCustomer
  動作： 消費者 ID を利用してデータベースからプロフィール情報を取得する.
  引数： int customer_id
  返り値： Customer customer

- updateProfile
  動作： データベース上で消費者の情報を更新する.
  引数： Customer customer
  返り値： なし

6.2.11 CourierHomeActivity クラス

配達員側のホーム画面を定義するクラスで，HomeActivity クラスを継承します．配達物の一覧を時間順や距離順でソートする機能を持ちます．配達員側の画面では，マップ画面に遷移することができます．CourierHomeActivity クラスのメソッドは以下の通りです.

- editProfile
  動作： courier から引数を取得する．また，getProfileCourier と updateProfile のメソッドを呼び出して変更を行う.
  引数： Courier courier
  返り値： なし

- getProfileCourier
  動作： 配達員 ID を利用してデータベースからプロフィール情報を取得する.
  引数： int driver_id
  返り値： Courier courier

- updateProfile
  動作： データベース上で配達員の情報を更新する
  引数： Courier courier
  返り値： なし

- getDeliveries
  動作： 配達員 ID を用いてデータベースから配達物一覧の取得を行う.
  引数： int driver_id
  返り値： ArrayList<Delivery> deliveriesInfo

- sortTimeButton
  動作： 配達物を時間順でソートするためのボタン．ソートする必要があるのか判定し，ソートを行う際は，sortTime メソッドを呼び出す．ソートが完了した配列を deliveriesInfo に格納し，refresh メソッドを呼び出す．
  引数： View view
  返り値： なし

- sortTime
  動作： deliveriesInfo(time) を利用し，時間順に配達物をソートする．
  引数： ArrayList<Delivery> deliveriesInfo
  返り値： なし

- sortDistanceButton
  動作： 距離順でソートするボタン．ソートする必要があるのか判定した上で，ソートを行う際は sortDistance を呼び出す．ソートが完了した配列を deliveriesInfo に格納し，refresh メソッドを呼び出す．
  引数： View view
  返り値： なし

- sortDistance
  動作： 距離順に配達物をソートする．商品の宛先情報から距離を計算する．
  引数： ArrayList<Delivery> deliveriesInfo
  返り値： なし

- showMapActivity
  動作： マップ画面へ遷移する．
  引数： Delivery delivery
  返り値： なし

6.2.12 DeliveryDetail クラス

配達物詳細画面を定義する継承元のクラスで，消費者側・配達員側の共通機能を実装します．荷物情報の詳細を表示する画面です．消費者側・配達員側の非共通機能はそれぞれのクラスで実装します．

- reschedulingButton
  配達日時の変更ボタンを押したときに発火される abstract メソッド．

6.2.13 CustomerDeliveryDetail クラス

消費者側の配達物詳細画面のクラスで，DeliveryDetail クラスを継承します．このクラスでは，荷物受取可否を選択する機能を実現します．CustomerDeliveryDetail クラスのメソッドは以下の通りです．

- reschedulingButton
  動作： 日時変更画面へ遷移する．
  引数： Delivery delivery
  返り値： なし

- receivableButton
  動作： 荷物を受け取るボタンを押したときに発火されるメソッド．荷物を受け取れる旨をサーバに送信する．
  引数： なし
  返り値： なし

6.2.14 CourierDeliveryDetail クラス

配達員側の配達物詳細画面を定義するクラスで，DeliveryDetail クラスを継承します．このクラスでは，配達完了を報告する機能を実装します．CourierDeliveryDetail クラスのメソッドは以下の通りです．

- reschedulingButton
  動作： 日時変更画面へ遷移する．
  引数： Delivery delivery
  返り値： なし

- deliveredButton
  動作： 配達完了ボタンを押したときに発火されるメソッド．配達完了の旨をサーバに送信する．
  引数： なし
  返り値： なし

6.2.15 TimeChange クラス

日時変更画面のクラスで，配達日時の変更を確定する機能を持ちます．

- createTimeChange
  動作： 変更確定ボタンを押したときに発火されるメソッド．日時の変更を確定する．変更内容をサーバに送信する．
  引数： なし
  返り値： なし

6.2.16 CourierMapActivity クラス

マップ画面を定義するクラスで，配達員側だけに表示される画面です．ピンの位置で配達先の住所を表示する機能や，受取可否ステータス別の絞り込み表示機能を実現します．CourierMapActivity クラスのメソッドは以下の通りです．

- setDeliveries
  動作： 配達物のリストを要求し格納する.
  引数： なし
  返り値： なし

- refineDeliveriesButton
  動作： 荷物情報の receivable_status を見て絞りこみ条件に一致するものの visible を true にし, それ以外を false にする.
  引数： View view
  返り値： なし

- showDeliveryDetailActivity
  動作： マップ画面から配達物詳細画面へ遷移する.
  引数： Delivery delivery
  返り値： なし

- refresh
  動作： マップ画面の再描画を行う.
  引数： なし
  返り値： なし

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. Server モジュールの引数や返り値で使用している単語は，表 8 の通りです. receivable_status は，未配達を 0，配達済みを 1 で表します. delivered_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 8: 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	customer_id	消費者の ID
int	manager_id	管理者の ID
int	id	配達員 ID，または，消費者 ID，または，管理者 ID
string	name	名前
string	password	パスワード
string	mail	メールアドレス
string	address	住所
long	tel	電話番号
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	delivered_status	受け取り可否
string	token	端末登録トークン

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： int id(customer_id または driver_id または manager_id)，String password
  返り値： int id(customer_id または driver_id または manager_id) (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： String name，String mail，String password，long tel，String 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： int driver_id
  返り値： String name，long slip_number，long ship_from，String address，int time，int receivable_status，int delivered_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： String name，string mail，String password，long 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： long slip_number，int time
  返り値： int 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： long 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： int driver_id
  返り値： String name，string mail，long 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： int customer_id
  返り値： String name，long slip_number，long ship_from，string address，int time，int receivable_status，int delivered_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： String name，string mail，String password，long tel，string 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： long 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： long slip_number，int time
  返り値： int 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： int customer_id
返り値： String name，string mail，long tel，string 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 (テキスト内容) を送信する.
  引数： String token
  返り値： String token，テキスト内容

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： int manager_id，String password
  返り値： なし

8 データベース設計

本システムのデータベースでは，Amazon RDS に MySQL を導入することで構築を行っています.

実体関連を表した ER 図を，図 4 に示します．PK は主キー，MK は外部キーを表しています.

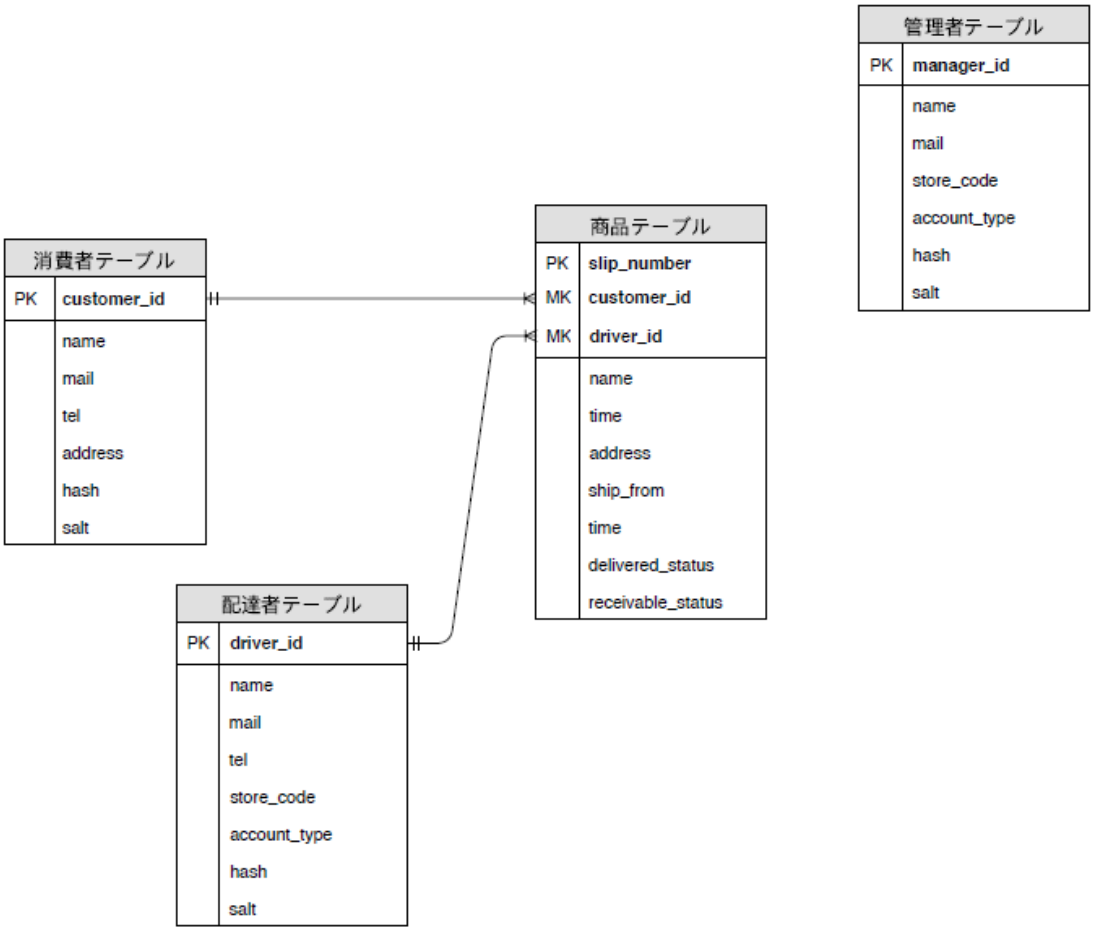

図 4: 実体関連図

8.1 各テーブルの詳細

本システムのデータベースには，4 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 9 に示します．

表 9: 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	bigint(11) unsigned	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 10 に示します．

表 10: 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	bigint(11) unsigned	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 11 に示します．

表 11: 商品テーブル

属性	データ型/長	NULL	Key	初期値	その他
slip_number	bigint(12) unsigned	NO	PRIMARY	NULL	
name	varchar(64)	NO			
address	varchar(128)	NO			
ship_from	bigint(64) unsigned	YES		NULL	
time	varchar(10)	YES		NULL	
delivered_status	int(1) unsigned	NO		0	
receivable_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 12 に示します．

表 12: 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			

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