社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

†

あらまし
キーワード

†

Abstract
Key words

1. はじめに

2. 準　　備

3. 提案手法

3.1 システム概要
3.2 ユースケース
3.3 システムアーキテクチャ
3.4 ドメインモデリング

ShelterNavi におけるドメインモデル図 1 にを示す．ShelterNavi には，図 1 で表されている 5 つのドメインが存在し，それぞれ Shelter（避難所），User（住民・ユーザ），Check-In（避難所へのチェックインまたはチェックアウト），ShelterState（避難所の状態），UserState（ユーザの状態）となっている．Check-In ドメインを通じてユーザとそのユーザが利用した避難所を関連付けさせるために，Check-In ドメインは必ず 1 つの User ドメインと 1 つの Shelter ドメインと結びつく．また，ShelterState ドメインは各避難所の状態を表しており，UserState ドメインはユーザの避難状態を示している．以下では，ShelterNavi の各ドメインを構成する要素について説明する．Shelter ドメインにおいては，システムがどの避難所か特定するための ID が必要であり，避難するユーザに向けて避難所の名称，位置情報を示す必要がある．そして，避難所の開放情報も取り扱う．以上のことから Shelter ドメインでは以下のスキーマを規定する．

- **sid**：避難所 ID
- **name**：避難所の名称
- **address**：避難所の住所
- **lat**：避難所の緯度
- **lng**：避難所の経度
- **capacity**：避難所の収容可能人数
- **isActive**：避難所が利用可能か

続いて User ドメインにおいては，ユーザ情報を一意に取り扱うためのユーザ ID が必要である．また，本サービスに登録するための email アドレスとパスワード，（名前，住所：必要性をどう示すか），そして混雑度を計算するための世帯人数を取り扱う．以上のことから User ドメインでは以下のスキーマを規定する．

- **uid**：ユーザ ID
- **email**：メールアドレス
- **password**：パスワード
- **name**：ユーザの名前
- **address**：ユーザの住所
- **of families**：世帯人数

Check-In ドメインにおいては，どのユーザがどの避難所を利用しているかを管理するためにユーザ ID と避難所 ID が必要である．また，複数の避難所への多重チェックインがないかを確認するために，チェックイン時刻と，チェックアウト時刻も取り扱う．以上のことから Check-In ドメインでは以下のスキーマを規定する．

- **cid**：チェックイン ID
- **uid**：ユーザ ID
- **sid**：避難所 ID
- **checkin-datetime**：チェックイン時刻
- **checkout-datetime**：チェックアウト時刻

ShelterState ドメインにおいては，コロナ禍における避難所での密を考慮すべく，避難所の混雑度も取り扱う．この混雑度は，避難所の収容可能人数に対しての現在の収容人数の割合から求められるので収容人数も管理する．そしてチェックイン，チェックアウトによる収容人数の変化を見るために，避難所の状態の最終更新日時も取り扱う．以上のことから ShelterState ドメインでは以下のスキーマを規定する．

- **id**：避難所の状態 ID
- **sid**：避難所 ID
- **num of accomodated**：収容人数
- **congestion**：混雑度
- **changedAt**：更新日時

UserState ドメインでは，ユーザの避難状態の履歴を管理する．特定のユーザの履歴を見るためにユーザ ID が必要である．そして，準備中，避難済み等の避難状態を表すスキーマを取

り扱い，避難状態が変化した際の日時を記録する．なお，このUserState ドメインでは過去のデータを参照できるようにしておくために，1 つのデータを更新していくのではなく，新規データを追加していく形をとる．

- **id**：ユーザの状態 ID
- **uid**：ユーザ ID
- **state**：避難状態
- **changedAt**：更新日時

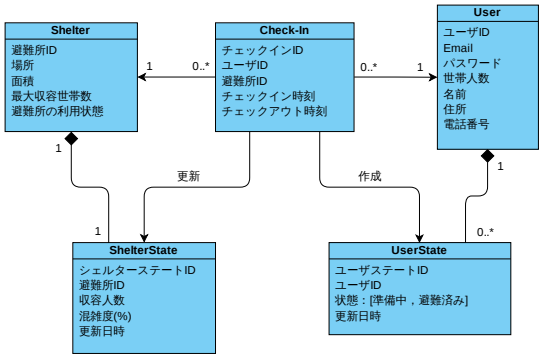

図 1　ドメインモデル図

3.5　主要なサービス

ShelterNavi では他のアプリケーションとの連携や拡張性を考慮し，HTTP を介して外部から利用できる API を配備した．以下に API の詳細を示す．

3.5.1　シェルターサービス

- **createShelter(shelterForm)**：避難所 ID，避難所名，位置情報を基に避難所データを作成し，取得する．
- **getShelter(sid)**：避難所 ID を指定することで該当する避難所データを取得する．
- **deleteShelter(sid)**：避難所 ID を指定することで該当する避難所データを削除する．
- **clearAllShelters()**：全ての避難所データを削除する．
- **getAllShelters()**：全ての避難所データを取得する．
- **searchSheltersByDistance(lng, lat, distance)**：経度，緯度，そして距離を指定することで，指定位置座標 (lng, lat) から半径 distance[km] 以内にある避難所データ全てを取得する．当 API では，地球上における大圏（大円）距離を計算する手法を使用しており，地球の半径を 6371[km]，ユーザの経度座標を lng，緯度座標を lat，各避難所データの経度座標を s_lng，緯度座標を s_lat とし，ユーザから半径 distance[km] 以内に存在する避難所を検索するものとして以下の計算式を用いる．

$$6371 \arccos(\cos(radians(lat)) \times \cos(radians(s\_lat)) \times \cos(radians(s\_lng) - radians(lng)) + \sin(radians(lat)) \times \sin(radians(s\_lat))) \leqq distance$$

- **searchSheltersByKeyword(keyword)**：文字列を指定することで，全避難所データの避難所名，または避難所の住所に部分一致するものがないか検索し，該当するデータを全て取得する．

3.5.2　ユーザサービス

- **createUser(userForm)**：メールアドレス，パスワード，世帯人数等のユーザ情報を元に新規アカウントを作成し，取得する．
- **getUser(uid)**：ユーザ ID を指定することで該当するユーザアカウントを取得する．
- **deleteUser(uid)**：ユーザ ID を指定することで該当するユーザアカウントを削除する．

3.5.3　チェックインサービス

- **checkIn(uid, sid)**：ユーザ ID と避難所 ID を指定することで，チェックインデータを作成し，更新日時を記録する．
- **checkOut(uid, sid)**：ユーザ ID と避難所 ID から一意に定まるチェックインデータを取得し，チェックアウト時刻を更新する．

3.5.4　シェルターステートサービス

- **checkIn(uid, sid)**：ユーザ ID と避難所 ID を指定することで，避難所の収容人数をユーザの世帯人数分増加させ，避難所の混雑度を計算し，更新する．また，更新日時を記録する．
- **checkOut(uid, sid)**：ユーザ ID と避難所 ID を指定することで，避難所の収容人数をユーザの世帯人数分減少させ，避難所の混雑度を計算し，更新する．また，更新日時を記録する．

3.5.5　ユーザステートサービス

- **checkIn(uid)**：ユーザ ID を指定することで，特定のユーザの避難状態を「避難済み」にした UserState のデータを新規で作成する．また，更新日時も併せて記録する．
- **checkOut(uid)**：ユーザ ID を指定することで，特定のユーザの避難状態を「準備中」にした UserState のデータを新規で作成する．また，更新日時も併せて記録する．

4.　実　　装

4.1　ShelterNavi プロトタイプの実装

今回は以下の開発環境で「ユーザサービス」,「シェルターサービス」，また「チェックインサービス」の一部の開発を行った．

- サーバ開発言語：Java
- クライアント開発言語：HTML5,JavaScript
- CSS ライブラリ：BootStrap
- データベース：MySQL 8.0.20
- Web サーバ：Apache Tomcat
- Web サービスフレームワーク：SpringBoot（Java）

以下では ShelterNavi の各画面構成に併せて実装の詳細を述べる．

4.2　サインアップ

4.3　ロ グ イ ン

今回の実装においては，セキュリティ性を担保するために Java のフレームワークである Spring Security を用いる．このフレームワークの機能を利用すれば，指定した URL 内で，ID とパスワードを Post する特定の API を利用することで認証が可能になる．また，ユーザオブジェクトに権限を付与し，その権限に応じた認可を与えることも可能になる．本アプリケーションでは，通常の工程でユーザを作成した場合，CITIZEN（住民・

一般ユーザ）の役割が付与される．実際には，ログイン機能における認証時にこの役割を見ることで，セッションに対して役割に対応した権限を付与する．これによりログイン後のユーザに対する各ページへの認可が可能になる．

4.4　避難所検索

4.5　チェックイン

本アプリケーションでは，避難所を可視化する上で Google Map を利用している．避難所を取得する API で避難所データを取得し，それらのデータを GoogleMapsAPI で利用することで，地図上への避難所の可視化を行っている．また，Shelter-Navi ではユーザの現在位置に応じて地図上に表示する避難所を変更しており，これは「3.5　主要なサービス」で言及した searchSheltersByDistance(lng, lat, distance) ををを利用している．

上記の計算により，ユーザから半径 distance[km] 以内の避難所を特定することが可能になる．

図 2　新規登録画面

図 3　ログイン画面

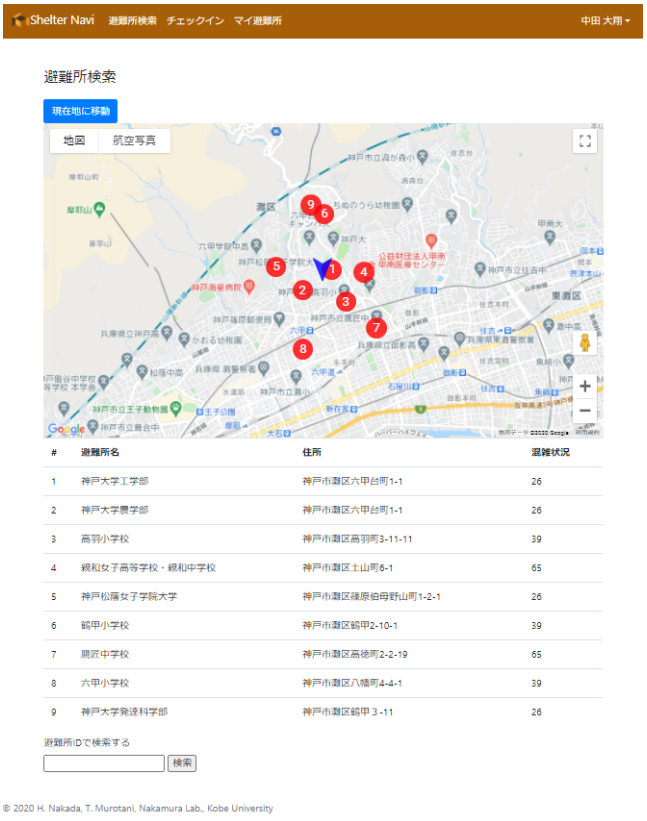

図 4　避難所検索画面

5.　考察・評価

6.　お わ り に

文　　献

[1]