情報システム論実習 (2017年度) オブジェクト指向計算演習（1）

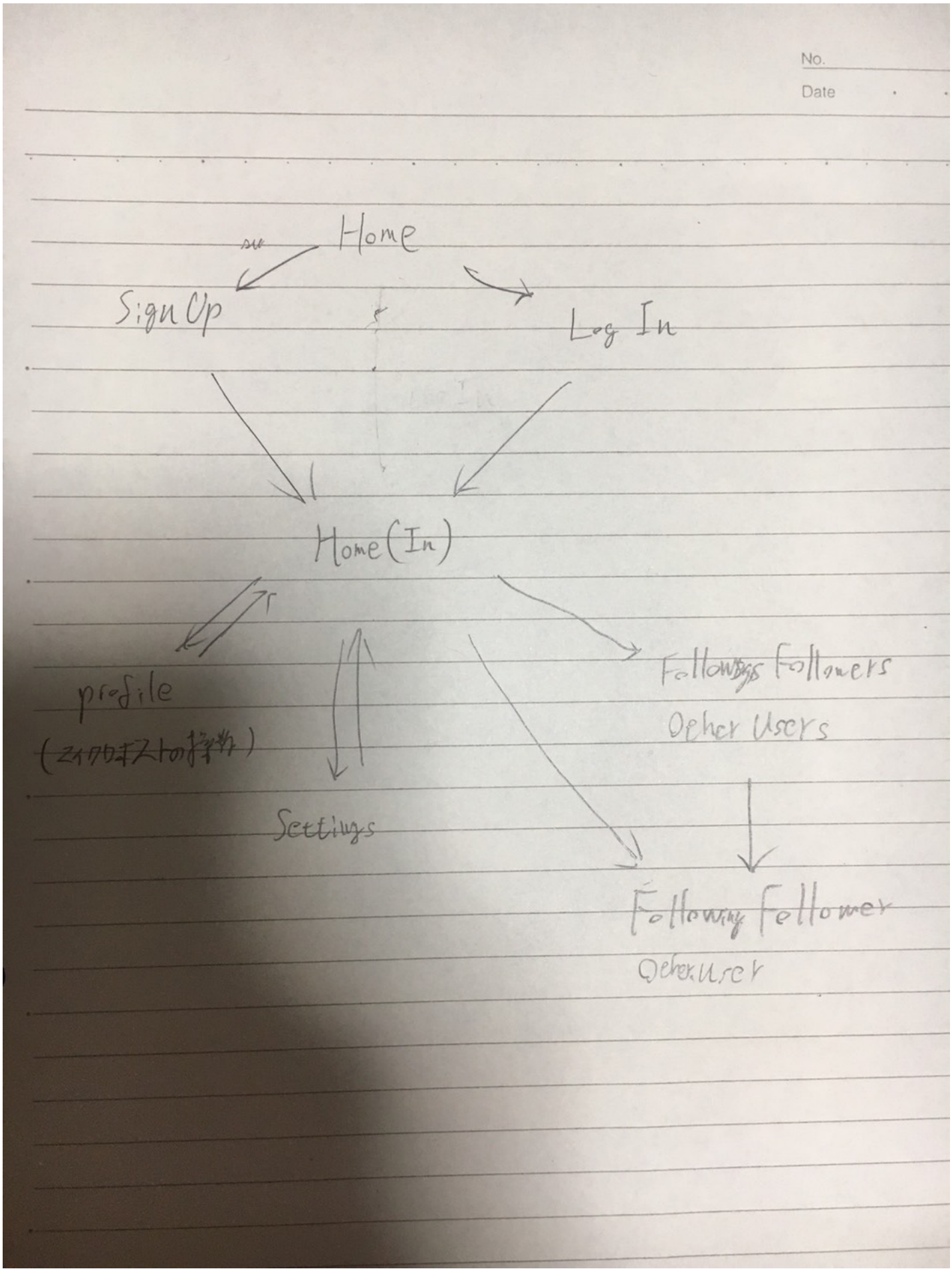
2017年7月9日
医療情報学講座
山崎陽平 (yohei7328@gmail.com)
6930-29-5923

全体説明

開発システムの概要

Twitterのような、投稿機能付きSNSアプリケーションを作成した。

システム構成


Home
Sign Up
Log In
Home(In)
profile
Settings
Followings Followers
Other Users
Following Follower
Other user

DBの構成

以下の三つのDBになる。

1. User
2. Micropost
3. Relationship

Userはユーサーの情報を格納するスキーマである。カラムとしては、以下の通り。

```
CREATE TABLE "users" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "name" varc
har, "email" varchar, "password_digest" varchar, "created_at" datetime NOT NULL, "
updated_at" datetime NOT NULL, "remember_digest" varchar, "admin" boolean DEFAULT
'f');
CREATE UNIQUE INDEX "index_users_on_email" ON "users" ("email");
```

Micropostはユーザの投稿したマイクロポストを格納するスキーマである。カラムとしては以下の通り。

```
CREATE TABLE "microposts" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "conte
nt" text, "user_id" integer, "created_at" datetime NOT NULL, "updated_at" datetime
 NOT NULL, "picture" varchar, CONSTRAINT "fk_rails_558c81314b"
FOREIGN KEY ("user_id")
  REFERENCES "users" ("id")
);
CREATE INDEX "index_microposts_on_user_id" ON "microposts" ("user_id");
CREATE INDEX "index_microposts_on_user_id_and_created_at" ON "microposts" ("user_i
d", "created_at");
```

Relationshipはユーザのフォロー・フォローされているを示した関係性のスキーマである。カラムとしては以下の通り。

```
CREATE TABLE "relationships" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, "fo
llower_id" integer, "followed_id" integer, "created_at" datetime NOT NULL, "update
d_at" datetime NOT NULL);
CREATE INDEX "index_relationships_on_follower_id" ON "relationships" ("follower_id
");
CREATE INDEX "index_relationships_on_followed_id" ON "relationships" ("followed_id
");
CREATE UNIQUE INDEX "index_relationships_on_follower_id_and_followed_id" ON "relat
ionships" ("follower_id", "followed_id");
```

公開先URL

http://yamasakiapp.herokuapp.com

利用シナリオ

1. ユーザー登録をする(既にユーザー登録している人はログインする)
2. マイクロポストの投稿を行う
3. Userから自分の好きなUserをフォローできる

アピールポイントとそのソースコード

1. bootstrapを用いたUI
2. フォロー・フォロワー実装のためのデータベースモデルの実装
3. Ajaxを用いたフォロー用UIの実装
4. 様々なvalidationの実装

1のソース

app/assets/javascripts/application.js `//= require rails-ujs //= require bootstrap`

2. フォローしているかどうかを以下のデータベースを設計することで実装した。

active_relationships	
follower_id	followed_id
1	2
1	7
3	1
7	2
1	10
2	1
1	8
9	1

3. Ajaxを用いてフォロー機能のUIを作成した。

```
class Relationship < ApplicationRecord
  belongs_to :follower, class_name: "User"
  belongs_to :followed, class_name: "User"
end
```

```
app/views/users/_follow.html.erb
<%= form_for(current_user.active_relationships.build, remote: true) do |f| %>
  <div><%= hidden_field_tag :followed_id, @user.id %></div>
  <%= f.submit "Follow", class: "btn btn-primary" %>
<% end %>

app/views/users/_unfollow.html.erb

<%= form_for(current_user.active_relationships.find_by(followed_id: @user.id),
             html: { method: :delete },remote: true
             ) do |f| %>
  <%= f.submit "Unfollow", class: "btn" %>
<% end %>
```

4. Userのvalidationを設定した。 具体的には

- ユーザ名が空にならない
- メールアドレスが被らない
- メールアドレスが重複しない
- パスワードが6文字以上 などである。

```
app/models/user.rb
class User < ApplicationRecord
  has_many :microposts, dependent: :destroy
  has_many :active_relationships, class_name:  "Relationship",
                                  foreign_key: "follower_id",
                                  dependent:   :destroy
  has_many :passive_relationships, class_name:  "Relationship",
                                   foreign_key: "followed_id",
                                   dependent:   :destroy

  has_many :following, through: :active_relationships, source: :followed
  has_many :followers, through: :passive_relationships, source: :follower

  before_save { self.email = email.downcase }
  validates :name,  presence: true, length: { maximum: 50 }
  # validates :email,  presence: true, length: { maximum: 255 } ,uniqueness: { cas
e_sensitive: false }
  # controllのテスト通らなくなるので一旦放置。
  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i
  validates :email, presence: true, length: { maximum: 255 },
                    format: { with: VALID_EMAIL_REGEX }, uniqueness: { case_sensit
ive: false }

  has_secure_password
  validates :password, presence: true, length: { minimum: 6 }, allow_nil: true
```

```
  def User.digest(string)
    cost = ActiveModel::SecurePassword.min_cost ? BCrypt::Engine::MIN_COST :
                                                  BCrypt::Engine.cost
    BCrypt::Password.create(string, cost: cost)
  end

  def feed
    following_ids = "SELECT followed_id FROM relationships
                     WHERE follower_id = :user_id"
    Micropost.where("user_id IN (#{following_ids})
                     OR user_id = :user_id", user_id: id)
  end

  def follow(other_user)
    active_relationships.create(followed_id: other_user.id)
  end

  def unfollow(other_user)
    active_relationships.find_by(followed_id: other_user.id).destroy
  end

  # 現在のユーザーがフォローしてたらtrueを返す
  def following?(other_user)
    following.include?(other_user)
  end

end
```

感想

普段最も使用するwebサービスがどのように動いているのかをコードを通して実際に知ることができた。