

採用面接課題

概要

写真管理アプリケーション（ツイート機能付き）の作成

条件

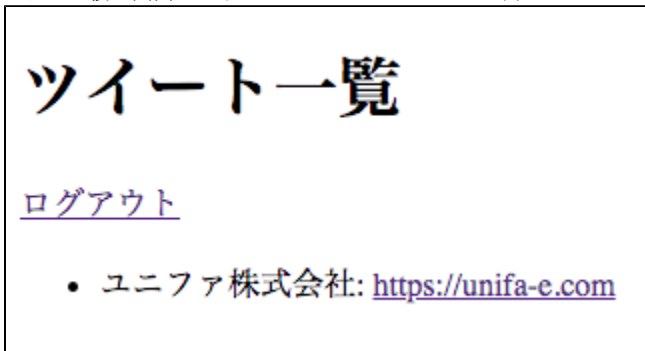
- 可能であれば「Ruby on Rails」を使用
 - 「Ruby on Rails」が使用できない場合、他のフレームワークでも可。
- 使用するプログラミング言語、フレームワークのバージョンは問わない。ただし、現在利用できるものにしてください。
- 可能な限りフレームワークに備わっているライブラリ以外は使用しない
 - Railsの場合、「rails new」して生成されるGemfileを変更しない。
 - ただし、コメントされているgemを使うのはOK。
 - 「slim」「haml」等、view templateに関するgemは自由に使ってOK。
 - とはいえ、時間も限られているので、ライブラリを使わないと時間がかかりすぎると判断した場合、使ってOKです。ただし、その場合はライブラリ内部でどういう処理を行なっているか、概要を質問する場合があります。
- gitでバージョン管理
- 要件をシンプルに実装してほしい。「今後こういった拡張があるかも」という想定は不要です。

事前確認

課題の後半で外部のアプリケーションとのOAuth連携を実装する箇所があります。その外部アプリケーションに関するチェックを事前に行ってください。

もし不備がある場合、弊社人事担当へご連絡ください。

- 課題と一緒に、OAuth連携テストアプリの以下の情報を受け取っていることを確認してください。
 - ユーザーID
 - パスワード
 - client_id
 - client_secret
- 以下のOAuth連携テストアプリにアクセスし、ログインページが表示されることを確認してください。（初回アクセスは表示に時間がかかる可能性あり）
<https://arcane-ravine-29792.herokuapp.com/>
- このログインページで「ユーザーID」「パスワード」を入力して、ログインできることを確認してください。
- ログイン後の画面が以下になっていることを確認してください。



課題詳細

以下の順序にしたがって、ローカル環境で動くWebアプリケーションを作成してください。

「画面例」は、要件を伝わりやすくする例です。要件が満たしてあれば、この画面通りになっていなくて構いません。

1. ログインページの作成

要件

- トップページにアクセスしたら表示
- 「ユーザーID」「パスワード」を入力して、ログインできる。
 - ユーザー登録は、事前にDBに直INSERTや、seedデータ等で実施している想定でOK
 - ここのログインは、こちらからお伝えした「ユーザーID」と「パスワード」は関係ありません。このアプリケーション独自でユーザーを管理できるようにしてください。
- ログインに成功した場合、次の2で行う写真一覧画面に遷移する。
- 以下の場合、ログインエラーとし、エラー原因と共にログイン画面を再表示する。
 - ユーザーIDが未入力
 - パスワードが未入力
 - ユーザーIDとパスワードが一致するユーザーが存在しない

画面例

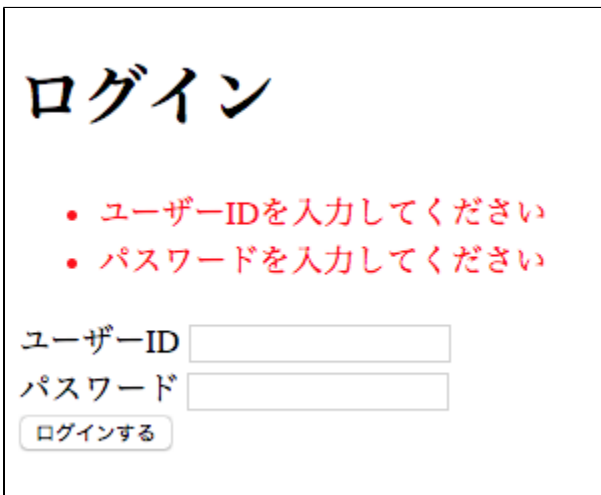


ログイン

ユーザーID

パスワード

エラー時の画面例



ログイン

- ユーザーIDを入力してください
- パスワードを入力してください

ユーザーID

パスワード

2. 写真一覧ページの作成

要件

- 現在ログイン中のユーザーがアップロードした写真を、最後にアップロードした順で表示する。
 - 表示する属性は「タイトル」「画像ファイル」
 - 最初のログイン直後は、アップロードした写真が一枚もないので、一枚も表示しないでOK
- 3で作成する「写真アップロード画面」へ遷移するリンクがある。
- 4で作成する「ログアウト」機能呼び出すリンクがある。
- 未ログイン状態でこの画面にアクセスした場合、ログイン画面を表示する。

画面例



3. 写真アップロード画面

要件

- 「タイトル」「画像ファイル」を指定して、画像をアップロードできる。
- 以下の場合、エラーメッセージと共にアップロード画面を再表示する。
 - 「タイトル」「画像ファイル」が未入力
 - 「タイトル」の文字数が30文字を超えている
- 画像アップロード処理が成功した場合、2の「写真一覧画面」を表示する。（アップロードした写真が先頭に表示されるはず）
- アップロードをキャンセルし、「写真一覧画面」に戻るリンクがある。
- 4で作成する「ログアウト」機能呼び出すリンクがある。
- 未ログイン状態でこの画面にアクセスした場合、ログイン画面を表示する。

画面例

[ログアウト](#)

写真アップロード

タイトル

画像ファイル No file chosen

[キャンセル](#)

エラー画面例

[ログアウト](#)

写真アップロード

- タイトルを入力してください
- 画像ファイルを入力してください

タイトル

画像ファイル No file chosen

[キャンセル](#)

4. ログアウト機能

ログイン後の画面にある「ログアウト」ボタンを押した時の処理を実装してください。

要件

- 未ログイン状態に戻る。
- ログアウト後、ログインページを表示する。

5. OAuth認可ページの表示

以降、外部にある架空のTwitterアプリケーションと連携する機能を実装します。

まず事前にOAuth連携テストアプリにログイン してください: <https://arcane-ravine-29792.herokuapp.com/>

この連携アプリは [OAuth 2.0 Authorization Code Flow](#) のフローにしたがった、API呼び出し用のアクセストークンを取得する機能を提供しています。

要件

- 写真一覧ページにOAuthの認可ページを開くためのリンクを表示。
- このリンクをクリックしたら、OAuth連携テストアプリの認可のページを開く。RFC6749の「(A)」の遷移になります。

リダイレクト用URLを作成するための情報は以下になります。

必要パラメーター	値
リダイレクト先URL	https://arcane-ravine-29792.herokuapp.com/oauth/authorize
client_id	弊社から事前にお伝えしている <code>client_id</code>
response_type	RFC6749の通り
redirect_uri	http://localhost:3000/oauth/callback 固定 もし他のURLに変更したい場合、変更したいURLと共に弊社人事部にご連絡ください。
scope	なし
state	なし

画面例

写真一覧画面の変更



認可ページの表示（この画面が表示できればOK。1度承認すると、2回目以降は省略されます）



6. OAuthアクセストークンの取得

5のページで「承認」ボタンを押すと、以下のURLにリダイレクトされ、自分のWebアプリケーションで認可コードを取得できます。

http://localhost:3000/oauth/callback?code=xxxxxxxx

このURLにパラメーターcodeとして付いている認可コードを使い、OAuth連携テストアプリケーションからアクセストークンを取得してください。

RFC6749の「(C)」～「(E)」の部分になります。

要件

- 「GET http://localhost:3000/oauth/callback」にリダイレクトされてもエラーにならないようにする。
- この中の処理でURLにセットされた認可コードからアクセストークンを取得し、セッションにセットする。
- アクセストークン取得後、写真一覧画面へリダイレクトさせる。
- 「否認」ボタンが押された時の処理は、対応しなくてOKです。

アクセストークンを取得する際に必要な情報は以下になります。

リクエスト

必要パラメーター	値
URL	https://arcane-ravine-29792.herokuapp.com/oauth/token
HTTP method	POST
code	RFC6749の通り
client_id	弊社から事前にお伝えしている <i>client_id</i>
client_secret	弊社から事前にお伝えしている <i>client_secret</i>
redirect_uri	http://localhost:3000/oauth/callback 固定 もし他のURLに変更したい場合、変更したいURLと共に弊社人事部にご連絡ください。
grant_type	RFC6749の通り

レスポンス

下記の情報がJSON形式でレスポンスBodyにセットされて返却されます。

パラメーター名	値
access_token	アクセストークンの値
その他	他にもいくつか属性が付いていますが、使うのはaccess_tokenだけのはず。

アクセストークンには有効期限はないので、一度取得すれば期限切れを気にせず使い続けてもらって構いません。

7. 連携アプリケーションヘツイートを投稿

連携アプリに画像のタイトルとURLでツイートを投稿する機能を実装してください。

要件

- セッションに連携アプリのアクセストークンがセットされている場合、各画像に「ツイートする」ボタンを表示
- ツイートするボタンを押した場合、以下の情報を含むツイートを連携アプリに作成する
 - 画像の「タイトル」
 - 画像を表示するための「URL」（imgタグに指定されているURL、localhost:3000のURLでOK）
- ツイート後、「写真一覧画面」ヘリダイレクト

連携アプリの「ツイート1件 作成」のAPIを呼び出すための情報は以下になります。

URL	https://arcane-ravine-29792.herokuapp.com/api/tweets
HTTP method	POST
HTTP header Content-Type	application/json
HTTP header Authorization	6で取得したアクセストークン。トークン種別は「Bearer」を使用してください。

Body	<p>以下のようなJSON文字列</p> <pre>{ "text": "", "url": "http://localhost:3000/photos/1.jpg" }</pre>
------	---

レスポンスは、作成したツイートがJSON形式で返ります。内容は重要ではなく、ステータスコードが201で返ってきていれば成功です。

画面例



ツイートするボタンを押したあと、連携アプリのログイン後画面を見てみてツイートした内容が表示されていれば成功です。(画像のURLは例です。作成するアプリケーションに合わせて変更してください)



課題の提出

- 以下のいずれかの情報を弊社人事担当者に送付してください。
 - githubにパブリックリポジトリとしてpushしたURL
 - gitのバージョン履歴を含むzipファイル

- READMEにコマンドの羅列で構いませんので、ローカルでの動かし方を記載してください。
- ざっくりで構いませんので、この課題にかけた時間を「xx時間」という形で、課題提出時に記載してください。
- その他何かありましたら、READMEに記載するか、課題提出時の連絡に記載してください。
- 課題が途中で構いませんので、納期優先で提出をお願いします。

例：

課題: <https://github.com/me/kadai>

かかった時間: 10時間