3072 新倉小雪　ブログシステムの概要

本システムの構成

・フレームワーク：Flask 2.3.3
・データベース：SQLite+SQLAlchemy ORM
・認証: Flask-Login + Werkzeug パスワードハッシュ化
・テンプレート: Jinja2

テーブル設計

ユーザーテーブル

id - 主キー
username - ユーザー名
email - メールアドレス
password_hash - ハッシュ化されたパスワード
created_at - アカウント作成日時

記事テーブル

id - 主キー
title - 記事タイトル（最大 200 文字）
content - 記事本文（テキスト形式）
author_id - 著者 ID（User テーブルへの外部キー）
is_published - 公開フラグ（デフォルト: True）
created_at - 作成日時（自動設定）
updated_at - 最終更新日時（自動更新）

コメントテーブル

id - 主キー
content - コメント本文（テキスト形式）
author_id - 投稿者 ID
article_id - 記事 ID
created_at - 投稿日時

返信テーブル

id - 主キー
content - 返信本文（テキスト形式）
author_id - 投稿者 ID
comment_id - コメント ID
parent_reply_id - 親返信 ID
created_at - 投稿日時

処理の流れ

アプリケーション起動処理

1. app.py 実行 → Flask アプリケーションインスタンス作成
2. データベース接続設定（SQLite）
3. Flask-Login 初期化
4. ルート定義の読み込み
5. データベーステーブル自動生成（db.create_all()）
6. サーバー起動（app.run()）

ユーザー登録処理

1. /register GET → 登録フォーム表示
2. ユーザー入力（username, email, password）
3. /register POST → バリデーション
   - メール重複チェック
   - ユーザー名重複チェック
4. パスワードハッシュ化（Werkzeug）
5. User レコード作成 → DB 保存
6. ログインページへリダイレクト

ログイン処理

1. /login GET → ログインフォーム表示
2. ユーザー入力（email, password）
3. /login POST → 認証処理
   - メールでユーザー検索
   - パスワード照合（check_password_hash）
4. 成功: login_user()でセッション作成
5. 失敗: エラーメッセージ表示

トップページへリダイレクト

④記事一覧表示処理

1. / GET → index()ルート呼び出し
2. Article.query.filter_by(is_published=True) → 公開記事のみ取得
3. order_by(created_at.desc()) → 新着順ソート
4. render_template('index.html', articles=articles) → テンプレートレンダリング
5. クライアントへ HTML 送信

記事作成処理

1. /create GET → 記事作成フォーム表示（login_required）
2. ユーザー入力（title, content, is_published）
3. /create POST → 記事保存処理
   - Article オブジェクト作成
   - author_id = current_user.id 自動設定
   - DB 保存（db.session.add() → commit()）
4. フラッシュメッセージ表示
5. トップページへリダイレクト

記事詳細表示処理

1. /article/<id> GET → article_detail()呼び出し
2. Article.query.get_or_404() → 記事取得
3. 権限チェック: 非公開記事は著者のみ閲覧可能
4. Comment.query.filter_by(article_id=id) → 関連コメント取得
5. render_template()で詳細ページ表示

記事編集処理

1. /edit/<id> GET → 編集フォーム表示（著者のみ）
2. 記事内容をフォームにプリフィル
3. /edit/<id> POST → 更新処理
   - 権限チェック（current_user.id == article.author_id）
   - フィールド更新
   - updated_at 自動更新
   - DB 保存

コメント投稿処理

1. /article/<id>/comment POST → add_comment()呼び出し
2. ログイン状態チェック（login_required）
3. コメント内容バリデーション
4. Comment オブジェクト作成
   - author_id = current_user.id
   - article_id = 対象記事 ID
5. DB 保存 → フラッシュメッセージ
6. 記事詳細ページへリダイレクト

返信投稿処理

1. /comment/<id>/reply POST → add_reply()呼び出し
2. ログイン状態チェック
3. 返信内容バリデーション
4. Reply オブジェクト作成
   - author_id = current_user.id
   - comment_id = 対象コメント ID
5. DB 保存 → フラッシュメッセージ
6. 記事詳細ページへリダイレクト

スレッド返信処理

1. /reply/<id>/reply POST → add_reply_to_reply()呼び出し
2. ログイン状態チェック
3. 返信内容バリデーション
4. Reply オブジェクト作成
   - parent_reply_id = 親返信 ID（スレッド構造）
5. DB 保存 → フラッシュメッセージ
6. 記事詳細ページへリダイレクト

テンプレートレンダリングフロー

1. ベーステンプレート（base.html）読み込み
2. ブロックコンテンツ置換
3. 動的変数展開（{{ variable }}）
4. 条件分岐（{% if %}）とループ（{% for %}）
5. URL 生成（{{ url_for('route_name') }}）

6. 静的ファイル参照（CSS, JS）

レスポンシブデザイン処理

1. メディアクエリによるデバイス判定
2. グリッドレイアウト自動調整
3. ナビゲーションのモバイル最適化
4. ボタンサイズの適応的変更

認証・認可フロー

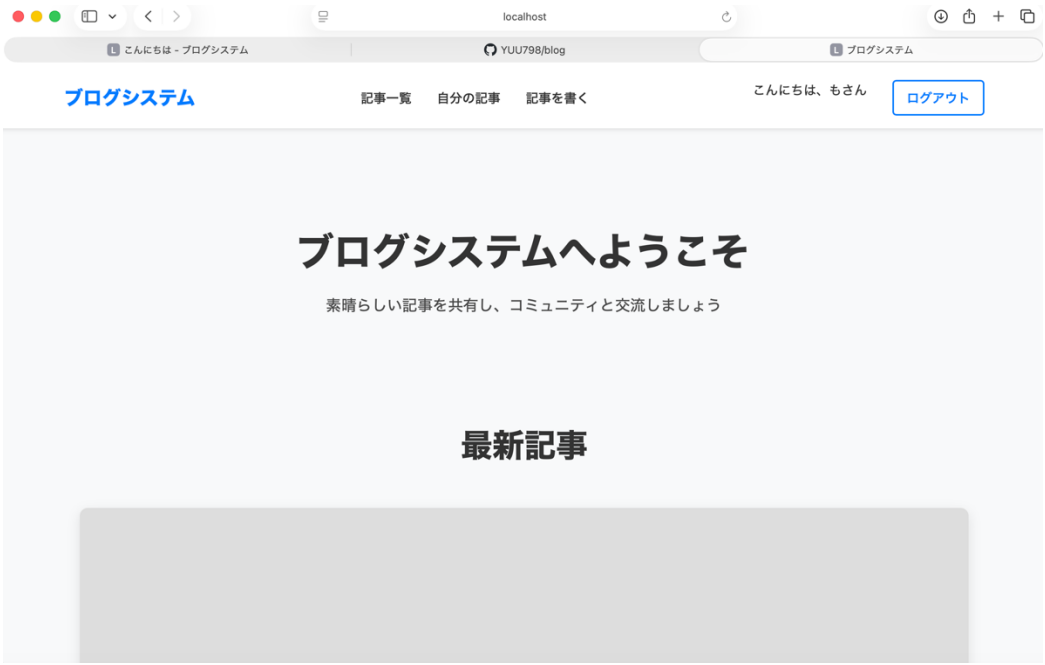
1. @login_required デコレータによるアクセス制御
2. セッションベース認証（Flask-Login）
3. パスワードハッシュ化（Werkzeug）
4. 記事編集/削除の権限チェック
5. 非公開記事のアクセス制限

エラーハンドリング

1. 404 エラー処理（get_or_404()）
2. データベースエラーキャッチ
3. フォームバリデーション
4. フラッシュメッセージによるユーザー通知

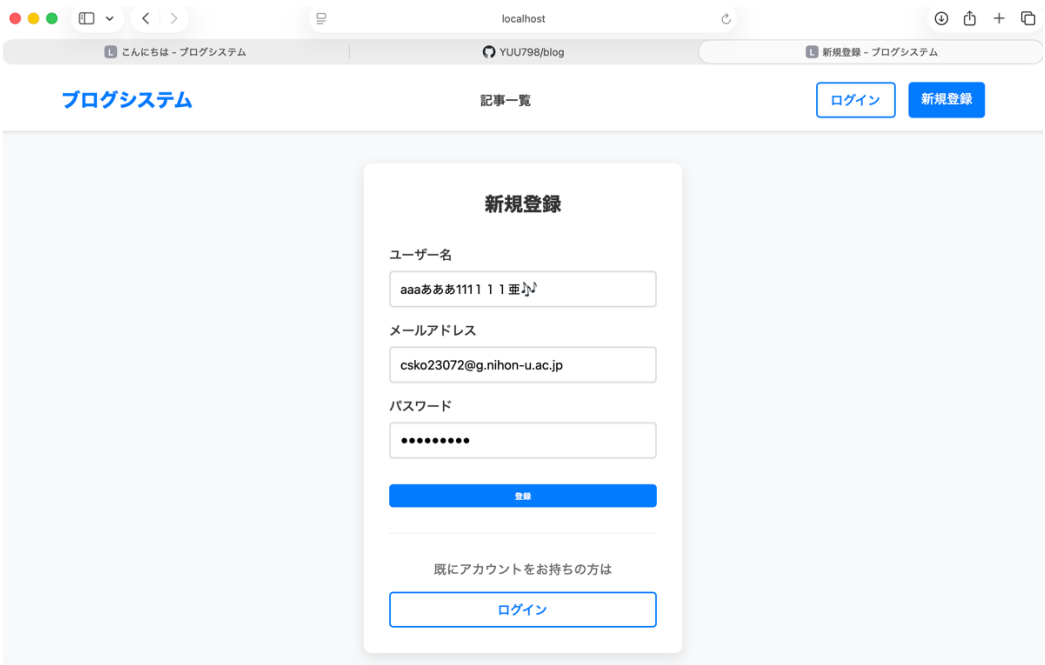
システムの動作画面

chmod +x start_production.sh && ./start_production.sh
python3 app.py
PORT=5002 python3 app.py(5002 は任意のポートで良い)
を実行した後に、http://localhost:5002(選択したポート番号)へアクセスするとホーム画面が表示される。



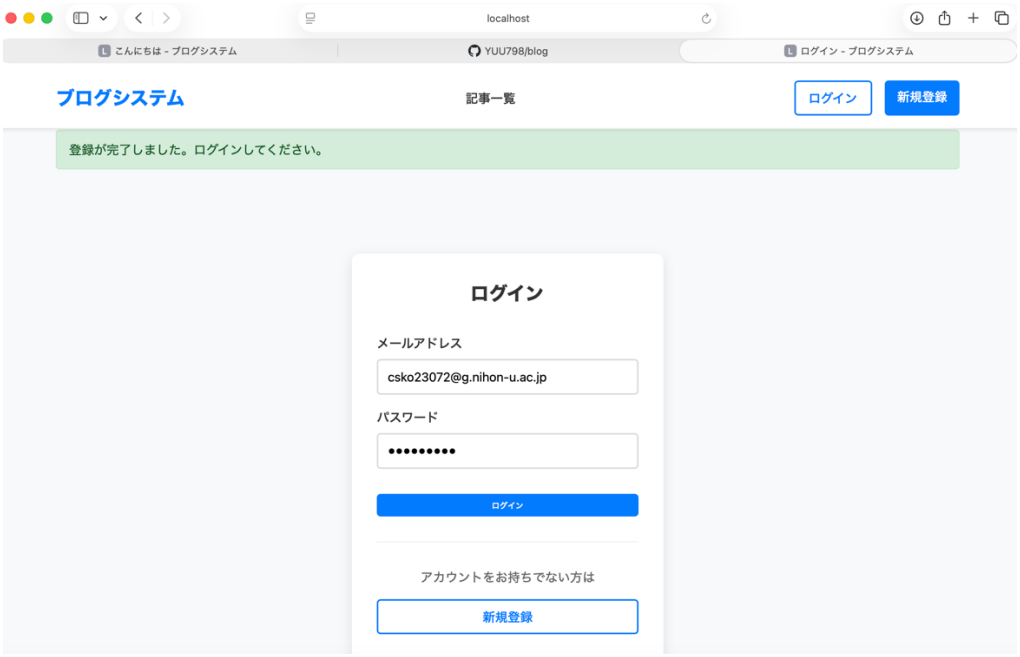
アカウントの登録・ログイン

画面右上にある新規登録(上画面ではすでにログインしているため、ログアウトが表示されている)を押すと次の画面に移りアカウントを登録することができる。

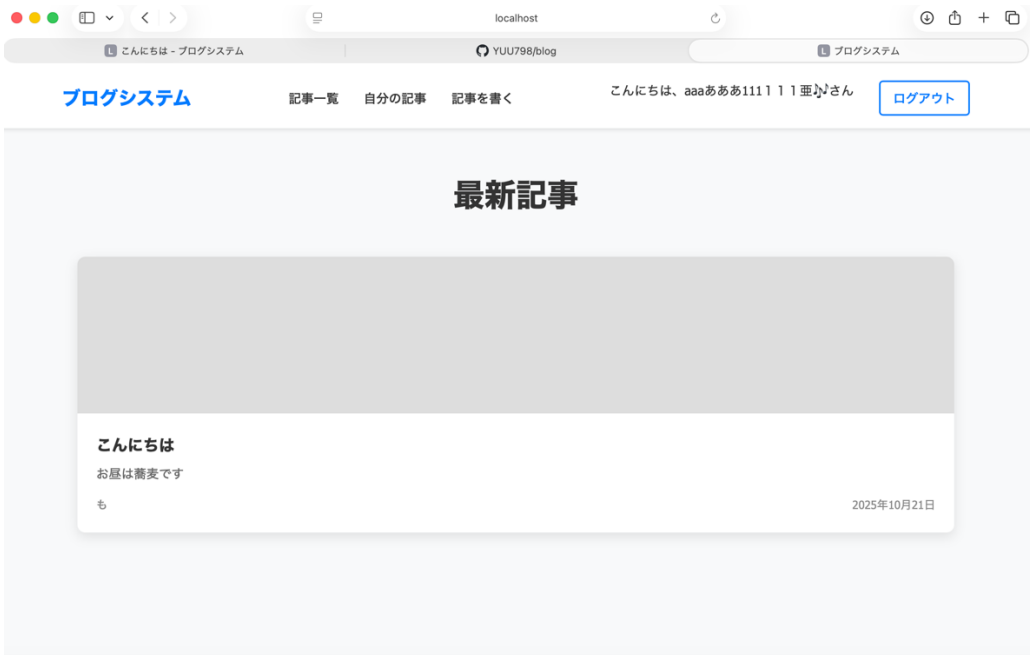


ユーザ名(英数字、平仮名、漢字、絵文字など可能)、メールアドレス、任意のパスワードを記入し中央の登録ボタンを押すとアカウントが作成できる。

登録すると自動的にログイン画面に移る。ログイン画面はホーム画面右上からも、新規登録ボタンの隣にあるログインボタンから移ることができる。先ほど登録したメールアドレスとパスワードを入力し、ログインボタンを押すとログインした状態でホーム画面が表示される。



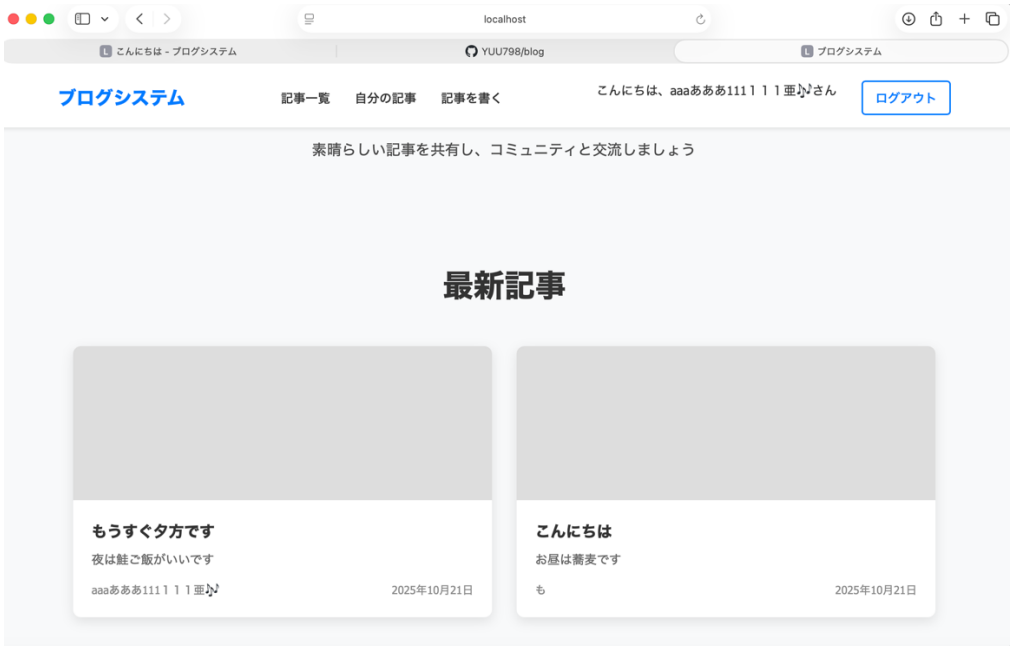
画面右上に登録したユーザ名が表示されて、ログインしている状態だと分かる。下にスクロールすれば他者のものも含めた投稿された記事を見ることができる。上の「記事を書く」から、自分も記事を投稿することができる。



記事の投稿

記事を書くを押すと次のような画面に移る。記事のタイトル、本文を入力することができる。また記事を公開するか非公開にするか選択することができる。チェックボックスを押して公開すると、先ほどのホーム画面に記事が表示される。一方でチェックボックスを外していると、ホーム画面には表示されない。

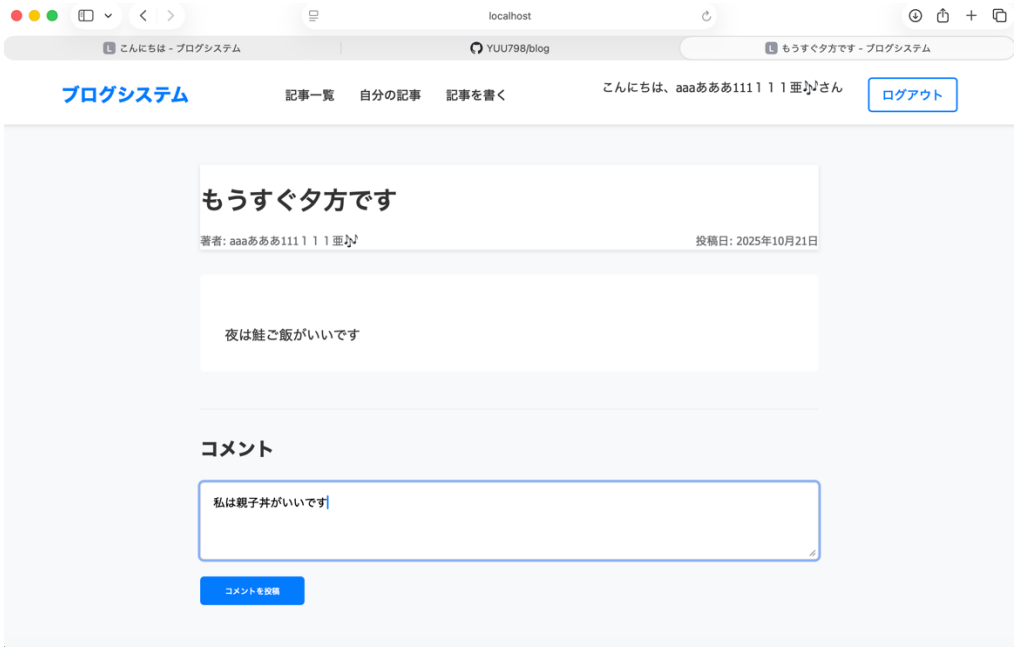
公開するとしたまま投稿するを押すと、ホーム画面に次のように反映される。一方で、公開するのチェックボックスを選択していないとホーム画面に記事は表示されない。



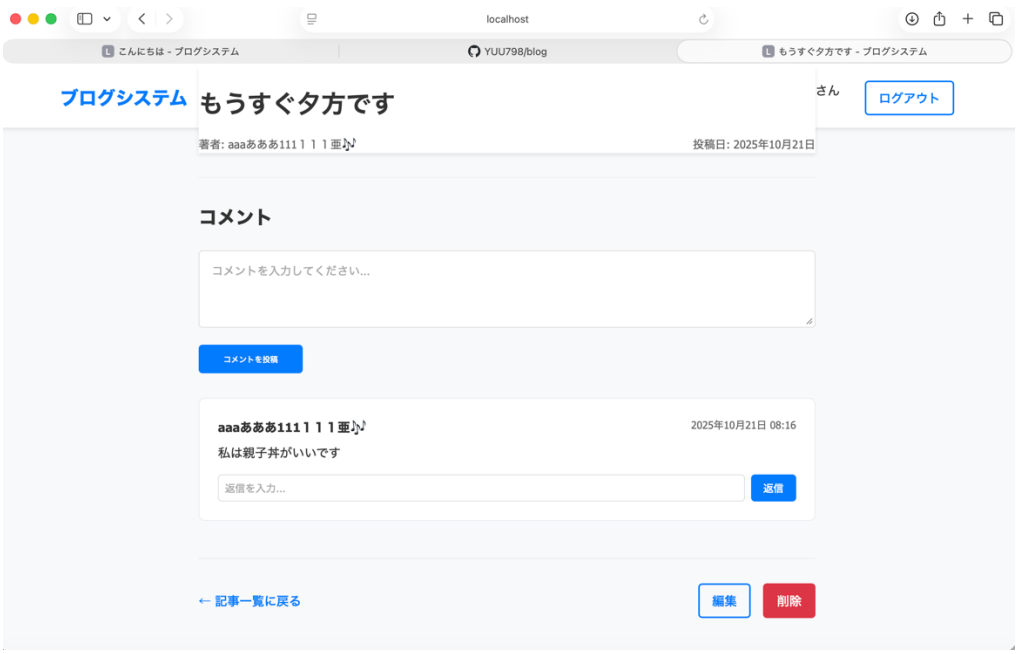
上記に表示されている各記事をクリックすると、記事の詳細を確認することができる。タイトル枠の右下で投稿された年月日を確認することができる。

コメント機能

記事に対してコメントすることができる。メント文を記入した後、コメント投稿ボタンを押す。



すると自動的に画面を更新し、コメントが反映される。コメント内にも返信する枠があり、コメントに対して返信を残すことができる。



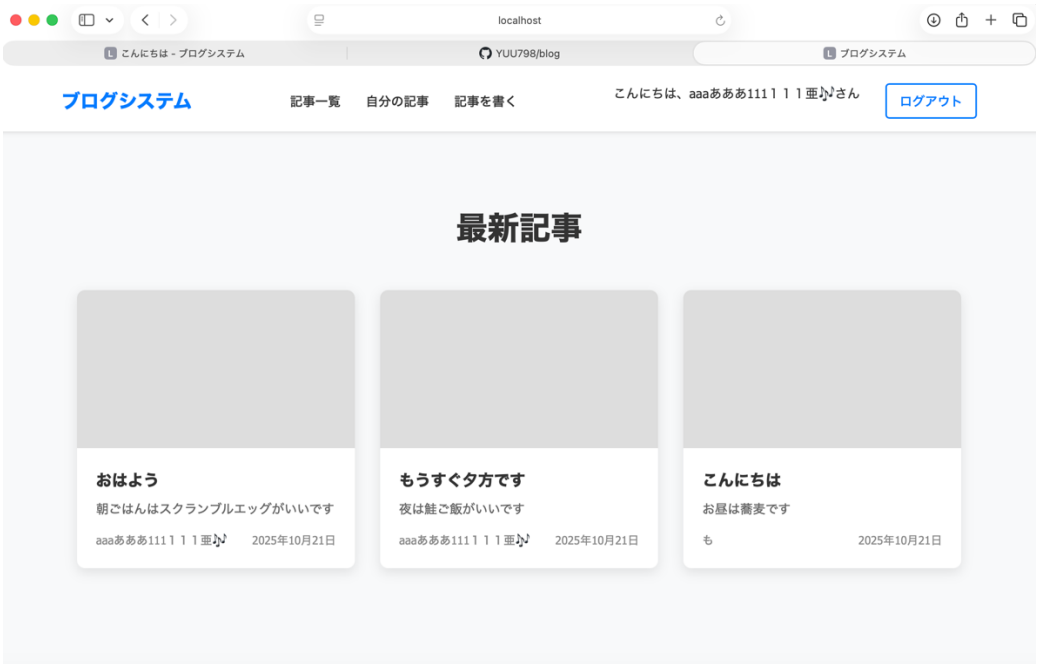
自分の記事

画面上にある記事を書くと並んで、自分の記事というボタンがある。これを押すことで自分が投稿した記事をまとめて確認することができる。実際に次のように記事が並んでいる。

この時、投稿した際に非公開となっていた記事もここで確認することができる。それぞれのフレームの右上に、非公開か公開か明記されている。

　また、各記事で右下にある編集を押すことで記事の編集ができる。更新するボタンを押すことで反映される。

テストで作成した「おはよう」という記事を試しに非公開から公開とすると、ホーム画面で表示されるようになった。



隣にある削除ボタンを押すと記事を削除することができる。1 度確認のポップアップが表示される。

ログアウト後

右上からログアウトできる。ログアウトした後でも自分の書いた記事は消されない。ログアウトした後は上のメニューが記事一覧だけとなっていて、あらゆる記事の編集や削除はできない。ログインした後でも以前に掲載した画面のように、他者の記事を編集することはできない。