

MemoLucky — Lucky For You

大学生向けキャンパスライフ支援プラットフォーム

luckyfouru

2442043 杉浦 芙美子、2442053 竹高 結衣、2442097 林 子嬌、2442103 小栗 花音

https://github.com/syokan00/Database_Final





Github



memolucky

課題背景とプロジェクトの目的



情報の非対称性

公式情報だけでは、研究室の雰囲気、就活の失敗談、単位取得のコツなど、「リアルな生の声」を得ることが困難です。



MemoLuckyの使命

経験の共有、フリマ機能、コミュニティ形成を通じて、孤独な意思決定をなくし、[学生生活の質\(QOL\)](#)を最大化します。

“

一人じゃない、仲間がいる安心感。

ターゲットユーザー（ペルソナ）



学部3年生：情報系

山田 太郎 さん

研究室選びに迷っている。「教授の指導は厳しい?」「コアタイムはある?」といった先輩の裏情報を求めている。メモ管理と出品を一つのUIで行いたい。



短大2年生：デザイン系

佐藤 花子 さん

実習道具を安く手に入れたい。授業ごとの詳細なメモを参考にしつつ、不要になった専門道具を後輩に譲りたい（フリマ活用）。

MemoLuckyの主要機能



匿名SNS機能

本名を明かさず、研究室や就活の本音を自由に投稿。心理的安全性を確保した設計。



検索

キーワードで細かく絞り込み。



フリマ・情報交換

教科書や実習道具の売買。経験に基づいた有益な情報には「役立った」評価が付与。



カスタム通知



貢献バッジ制度

信頼性の担保と安全設計

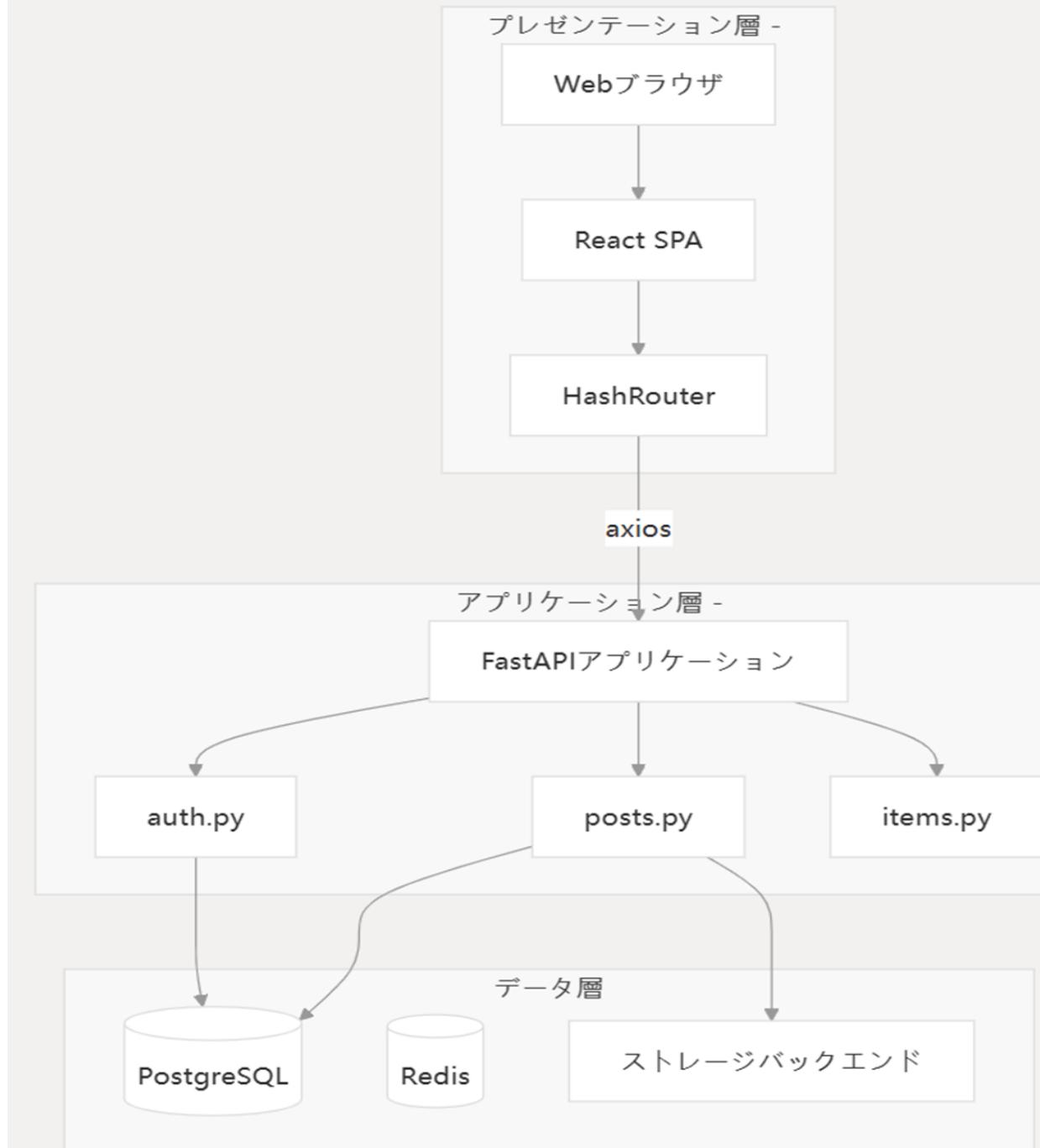
- JWTベースの認証フロー
 - 最新の暗号化技術により、個人情報とセッションを保護
 - 。
- 通報・評価システム
 - 不適切な投稿は即時排除。良質な貢献者にバッジを付与
 - 。



システム構成と技術スタック

3層アーキテクチャ

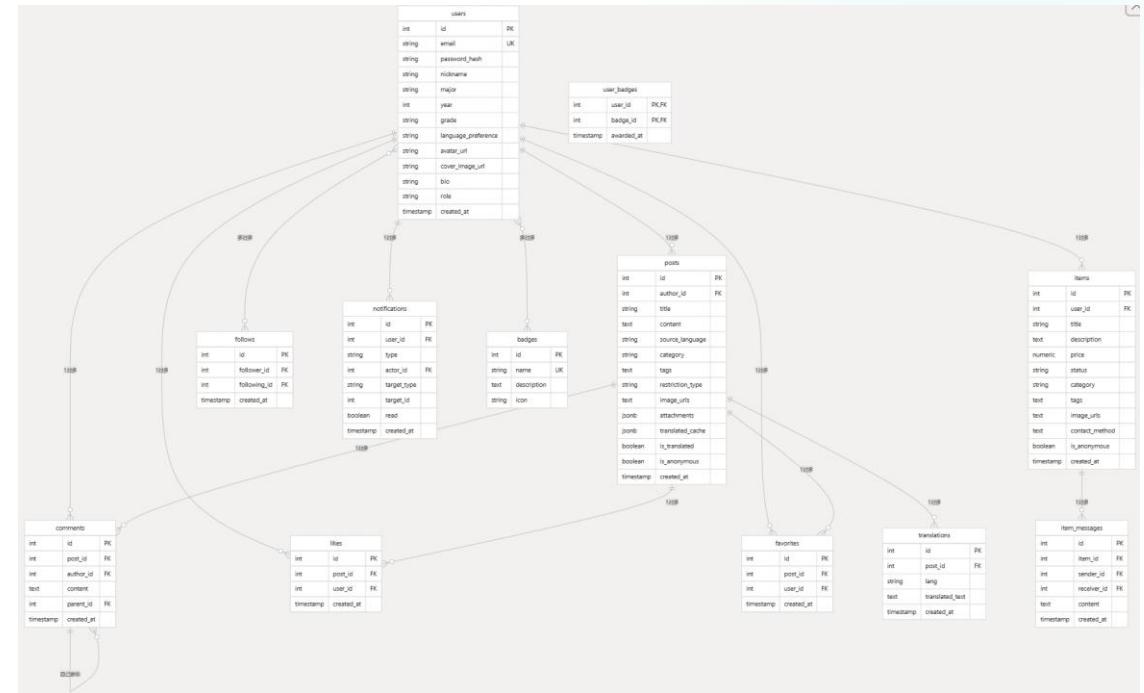
- 1 プrezentashon层: 直感的で使いやすいUI/UXを提供。学生が迷わず操作できるレスポンシブ設計。
- 2 アプリケーション层: 認証ロジック、検索アルゴリズム、通知エンジンを制御。
- 3 データ层: 大規模な投稿データやユーザー情報を構造化して安全に管理。



データベース設計 (ER図)

memoluck-> \dt

スキーマ	名前	タイプ	所有者
public	badges	テーブル	memolucky_user
public	comments	テーブル	memolucky_user
public	favorites	テーブル	memolucky_user
public	follows	テーブル	memolucky_user
public	item_messages	テーブル	memolucky_user
public	items	テーブル	memolucky_user
public	likes	テーブル	memolucky_user
public	notifications	テーブル	memolucky_user
public	posts	テーブル	memolucky_user
public	translations	テーブル	memolucky_user
public	user_badges	テーブル	memolucky_user
public	users	テーブル	memolucky_user
(12 行)			



完全ER図（分割表示）

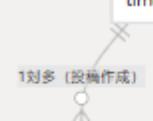
コアユーザー・コンテンツテーブル

users			
int	id	PK	主キー (Primary Key)
string	email	UK	メールアドレス (一意制約)
string	password_hash		パスワードハッシュ
string	nickname		ニックネーム
string	major		専攻
int	year		入学年
string	grade		学年
string	language_preference		言語設定
string	avatar_url		アバター画像URL
string	cover_image_url		背景画像URL
string	bio		自己紹介
string	role		ロール (user/admin)
timestamp	created_at		作成日時

posts			
int	id	PK	主キー
int	author_id	FK	投稿者ID (外部キー)
string	title		投稿タイトル
text	content		投稿内容
string	source_language		投稿言語
string	category		カテゴリ
text	tags		タグ
string	restriction_type		制限タイプ
text	image_urls		画像URL
jsonb	attachments		添付ファイル
jsonb	translated_cache		翻訳キャッシュ
boolean	is_translated		翻訳済みフラグ
boolean	is_anonymous		匿名投稿フラグ
timestamp	created_at		作成日時

items			
int	id	PK	主キー
int	user_id	FK	出品者ID (外部キー)
string	title		アイテム名
text	description		アイテム説明
numeric	price		価格
string	status		ステータス
string	category		カテゴリ
text	tags		タグ
text	image_urls		画像URL
text	contact_method		連絡方法
boolean	is_anonymous		匿名出品フラグ
timestamp	created_at		作成日時

1対多 (投稿作成)

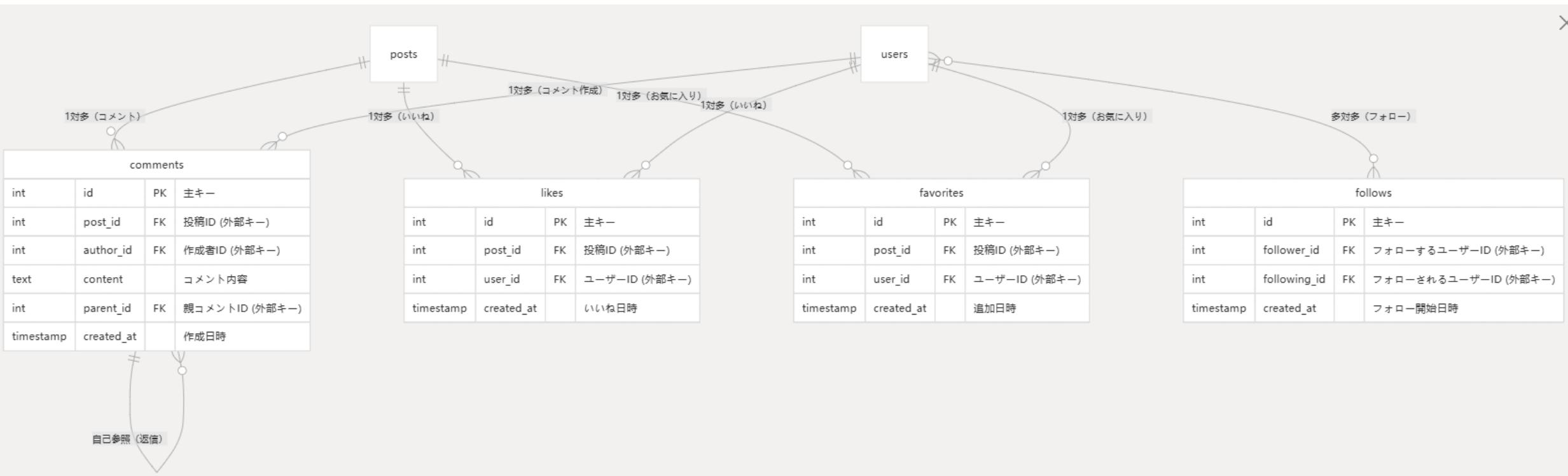


1対多 (出品)



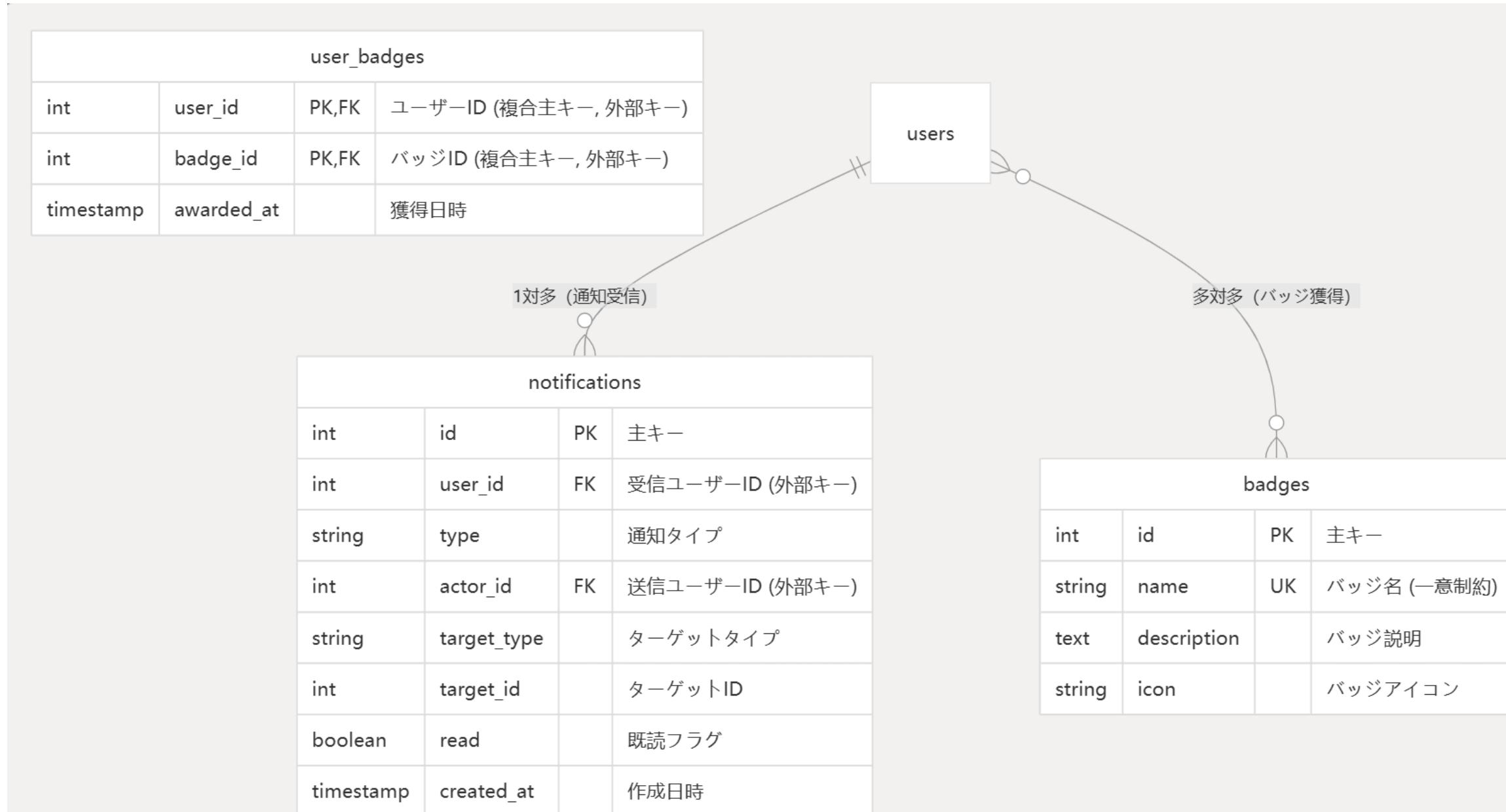
完全ER図（分割表示）

ソーシャルインテラクションテーブル



完全ER図（分割表示）

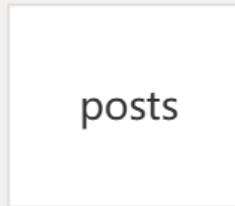
システム管理テーブル



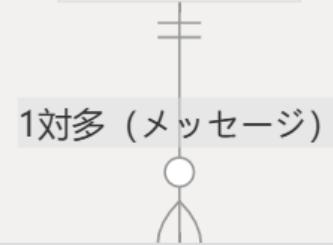
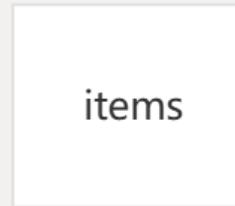
完全ER図（分割表示）

サポート機能テーブル

X



int	id	PK	主キー
int	post_id	FK	投稿ID (外部キー)
string	lang		翻訳言語
text	translated_text		翻訳テキスト
timestamp	created_at		作成日時



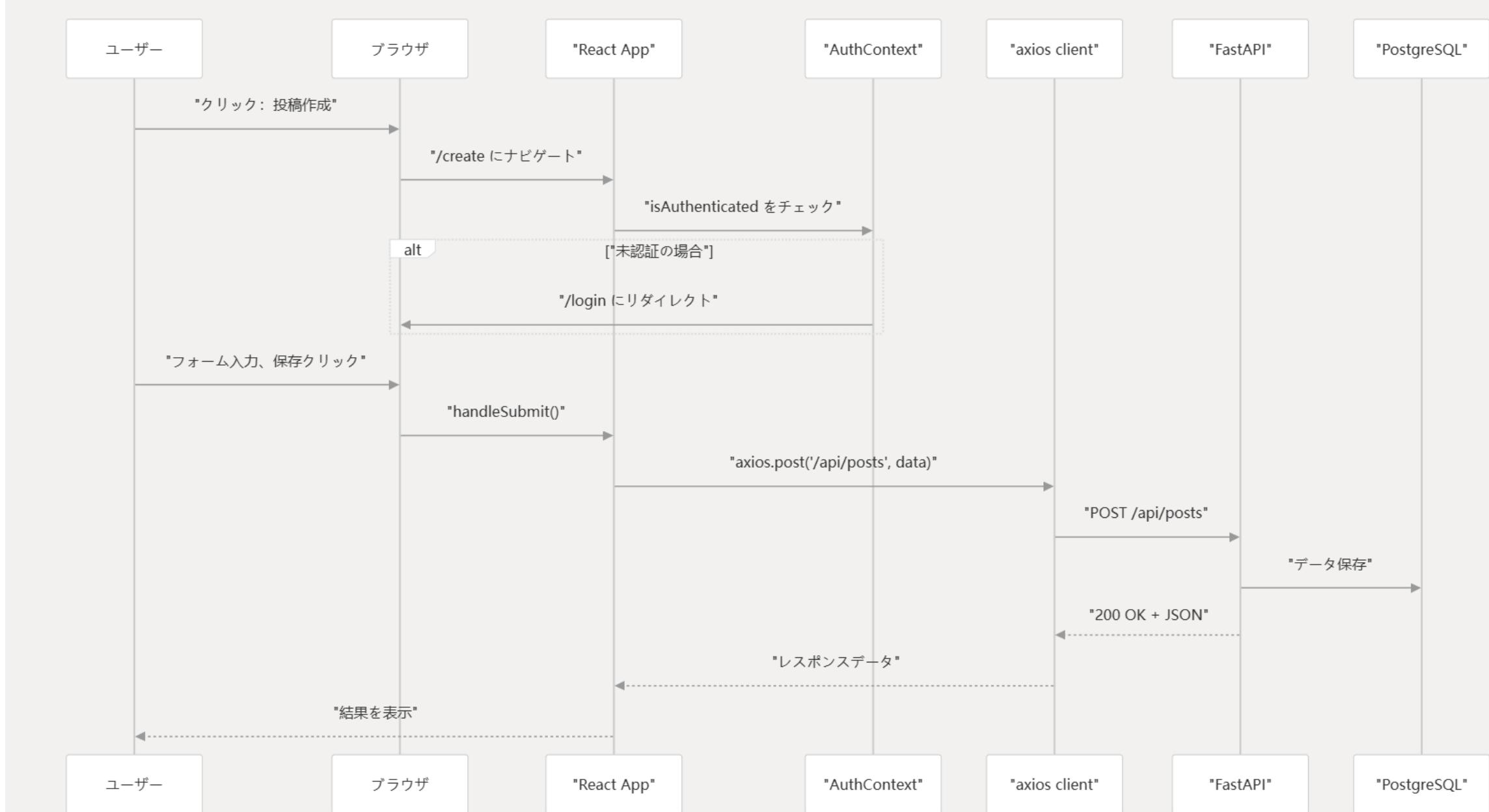
int	id	PK	主キー
int	item_id	FK	アイテムID (外部キー)
int	sender_id	FK	送信者ID (外部キー)
int	receiver_id	FK	受信者ID (外部キー)
text	content		メッセージ内容
timestamp	created_at		送信日時

認証・リクエスト処理フロー



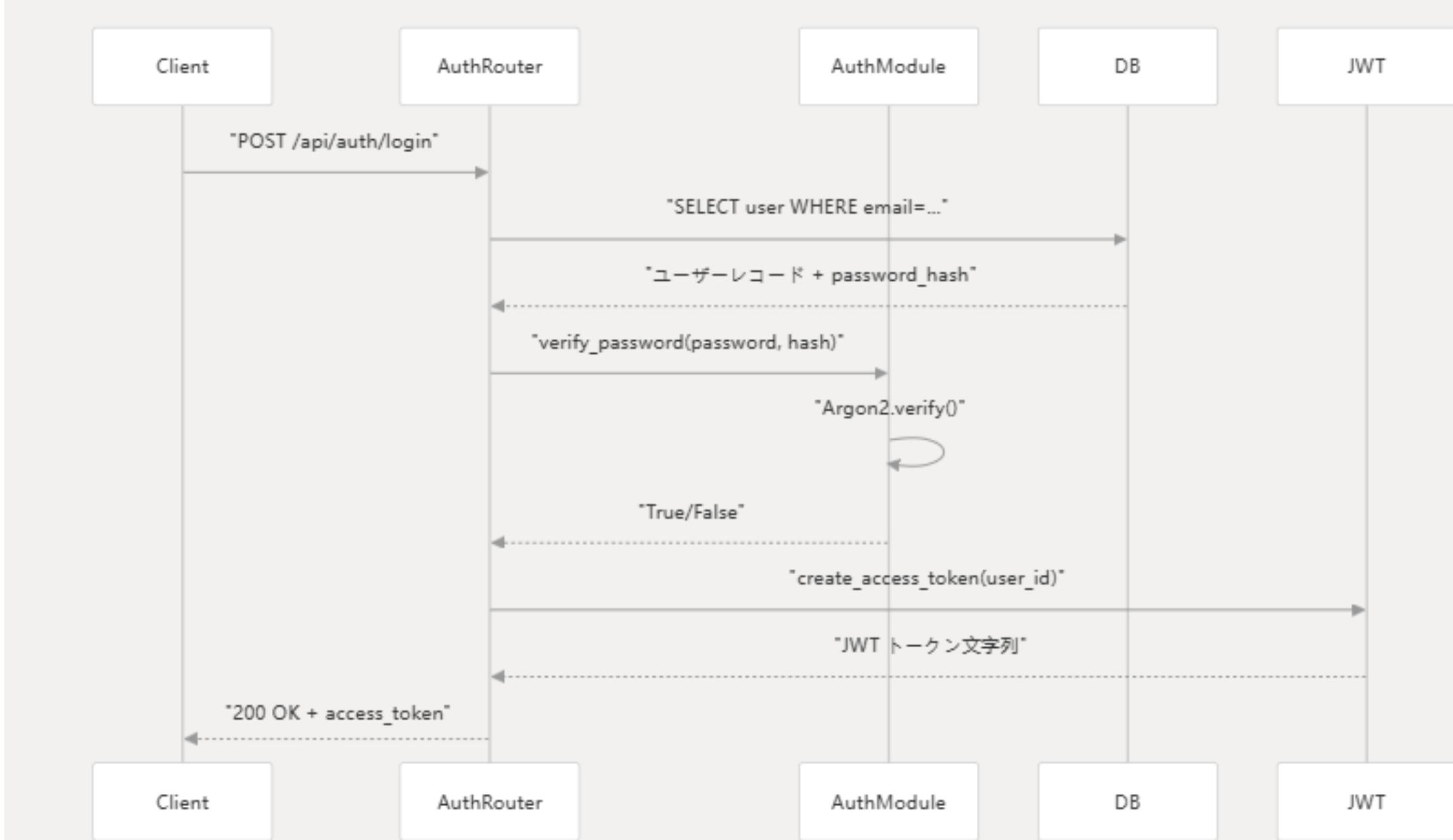
リクエストフロー図

目的: ユーザー操作からシステム応答までの完全な流れを示す



認証フロー図

目的: JWTベースの認証メカニズムを理解する



MemoLuckyがもたらす 価値

☒ 業務価値 / ROI

学生同士の情報共有を促進し、キャンパスコミュニティを活性化。ミスマッチな研究室選びやキャリア選択を防ぎ、[学生のキャリア価値を向上](#)させます。

🌐 拡張性

日本人学生だけでなく留学生とも交流しやすい環境を構築。大学全体の国際化と相互扶助の文化を醸成します。



機能実装状況まとめ

機能	実装状況	コード出処	備考
SQL	✓ 実装済み	db/init.sql, backend/app/*.py	-
トランザクション	✓ 実装済み	backend/app/auth.py:113, 124 backend/app/posts.py:175, 182	-
ストレージエンジン	✓ PostgreSQL デフォルト	backend/app/database.py:60	-
インデックス	✓ 実装済み	db/init.sql:106-108 backend/app/models.py:9-10	-
SQL 最適化	✓ 部分実装	backend/app/posts.py:68	-
ロック	✓ 自動処理	-	PostgreSQL MVCC
ログ	✓ アプリケーション層ログ	backend/app/auth.py:116, 126	-
レプリケーション	✗ 未実装	-	-
読み書き分離	✗ 未実装	-	-
シャーディング	✗ 未実装	-	-
関数	⚠ 組み込み関数のみ	db/init.sql:15, 108	-
制約	✓ 実装済み	db/init.sql:3-102	-
複数テーブルクエリ	✓ 実装済み	backend/app/models.py:23-69	-
ビュー	✗ 未実装	-	-
ストアドプロシージャ	✗ 未実装	-	-
トリガー	✗ 未実装	-	-
InnoDB	✗ 適用外	-	(PostgreSQL)
MySQL 管理	✗ 適用外	-	(PostgreSQL)

SQLステートメント分類別まとめ表

DDL (データ定義言語) - 合計 26

タイプ	数量	具体的なステートメント例	ファイル位置
CREATE TABLE	9	CREATE TABLE users, badges, posts, translations, comments, likes, items, user_badges, favorites	db/init.sql:2-103
CREATE INDEX	4	CREATE INDEX idx_posts_created, idx_posts_category, idx_posts_tags, idx_posts_attachments	db/init.sql:106-108, migrations/004_*.sql:9
ALTER TABLE	6	ALTER TABLE users ADD COLUMN grade/cover_image_url/bio, posts ADD COLUMN image_urls/attachments/is_anonymous	main.py:21-29, migrations/*.sql
その他 (CREATE DATABASE, USER, COMMENT)	7	CREATE DATABASE memoluck, CREATE USER readonly_demo, COMMENT ON COLUMN...	database.py:41, create_READONLY_user.sql:6, migrations/*.sql

DML (データ操作言語) - 合計 40+

タイプ	数量	具体的なステートメント例	ファイル位置	説明
INSERT (一括)	1	INSERT INTO badges VALUES (...) (14件)	db/init.sql:111-126	バッジデータ初期化
INSERT (ORM)	20+	ユーザー登録, 投稿作成, コメント作成, いいね, 收藏, 关注, メッセージ送信, 通知作成 等	全ての *.py	ORMによる自動生成
UPDATE (ORM)	10+	プロフィール更新, 投稿更新, 商品更新, パスワード変更, 通知既読化 等	auth.py, posts.py, items.py	ORMによる自動生成
DELETE (ORM)	8+	投稿削除, コメント削除, 商品削除, ユーザー削除, いいね取消, 收藏取消, 关注取消 等	posts.py, comments.py, items.py	ORMによる自動生成

DCL (データ制御言語) - 合計 6

タイプ	数量	具体的なステートメント例	ファイル位置
GRANT CONNECT	1	GRANT CONNECT ON DATABASE postgres TO readonly_demo	create_READONLY_user.sql:9
GRANT USAGE	1	GRANT USAGE ON SCHEMA public TO readonly_demo	create_READONLY_user.sql:12
GRANT SELECT	2	GRANT SELECT ON ALL TABLES...; ALTER DEFAULT PRIVILEGES... GRANT SELECT...	create_READONLY_user.sql:15, 18-19
GRANT SEQUENCE	2	GRANT USAGE, SELECT ON ALL SEQUENCES...; ALTER DEFAULT PRIVILEGES... GRANT USAGE, SELECT...	create_READONLY_user.sql:22-24

DQL (データ問い合わせ言語) - 合計 70+

タイプ	数量	具体的なステートメント例	ファイル位置
SELECT (直接)	2	SELECT 1; SELECT 1 FROM pg_database WHERE datname = 'database'	database.py:26, 39
SELECT (ORM)	50+	ユーザー照会, 投稿一覧, コメント照会, 商品照会, いいね状態, 收藏, 关注関係, 通知, メッセージ, 統計 等	全ての *.py
SELECT with JOIN	10+	posts JOIN users, comments JOIN users, user_badges JOIN badges, notifications JOIN users 等	posts.py, users.py
SELECT with COUNT	5+	SELECT COUNT(*) FROM follows/comments/items WHERE ...	users.py:90, badge_service.py:111
SELECT with EXTRACT	1	SELECT * FROM posts WHERE EXTRACT('hour', created_at) BE	badge_service.py:36

SQLステートメント統計総括表

SQLタイプ	数量	主なステートメント	説明
DDL	26	CREATE TABLE, INDEX, ALTER TABLE, etc.	データ定義言語
DML	40+	INSERT, UPDATE, DELETE (含ORM)	データ操作言語
DCL	6	GRANT CONNECT, USAGE, SELECT, SEQUENCE	データ制御言語
DQL	70+	SELECT (含JOIN, COUNT, EXTRACT, ORM)	データ問い合わせ言語
総計	142+		全SQLステートメント

トランザクション

トランザクション統計表 / Transaction Statistics Table

モジュール / Module	トランザクション数 Transaction Count	ロールバック有 With Rollback	ロールバック無 Without Rollback	複数テーブル操作 Multi-table Operation
認証モジュール / Authentication Module	3	1⚠️	2	0
投稿モジュール / Posts Module	5	1⚠️	4	1✅ (「いいね」 / Like Post)
コメントモジュール / Comments Module	2	0	2	1✅ (コメント作成 / Create Comment)
商品モジュール / Items Module	3	0	3	0
ユーザーモジュール / Users Module	3	0	3	1✅ (ユーザー フォロー / Follow User)
お気に入りモジュール / Favorites Module	2	0	2	1✅ (お気に入り追加 / Add to Favorites)
通知モジュール / Notifications Module	3	0	3	0
メッセージモジュール / Messages Module	1	0	1	1✅ (メッセージ送信 / Send Message)
ファイルアップロード / File Upload	2	0	2	0
バッジサービス / Badge Service	1	1⚠️	0	0
アプリケーション起動 / Application Startup	2	1⚠️	1	0
合計 / Total	27	4⚠️	23	5✅

全トランザクションリスト (ファイル別)

auth.py			
ID	行番号	操作	ロールバック
1	113	ユーザー登録	完了 ✅
2	191	プロフィール更新	未完了 ✗
3	205	パスワード変更	未完了 ✗

posts.py			
ID	行番号	操作	ロールバック
4	175	投稿作成	完了 ✅
5	294	投稿に「いいね」	未完了 ✗
6	306	通知作成	未完了 ✗
7	330	「いいね」取消	未完了 ✗
8	367	投稿更新	未完了 ✗
9	420	投稿削除	未完了 ✗

comments.py			
ID	行番号	操作	ロールバック
10	58	コメント作成	未完了 ✗
11	71	通知作成	未完了 ✗
12	110	コメント削除	未完了 ✗

items.py			
ID	行番号	操作	ロールバック
13	77	商品作成	未完了 ✗
14	111	商品削除	未完了 ✗
15	151	商品更新	未完了 ✗

favorites.py			
ID	行番号	操作	ロールバック
20	30	お気に入り追加	未完了 ✗
21	42	通知作成	未完了 ✗
22	62	お気に入り解除	未完了 ✗

notifications.py			
ID	行番号	操作	ロールバック
23	31	通知作成	未完了 ✗
24	92	既読にする	未完了 ✗
25	106	すべて既読にする	未完了 ✗

messages.py			
ID	行番号	操作	ロールバック
26	49	メッセージ送信	未完了 ✗
27	62	通知作成	未完了 ✗

uploads.py			
ID	行番号	操作	ロールバック
28	44	アバターアップロード	未完了 ✗
29	89	カバーアップロード	未完了 ✗

badge_service.py			
ID	行番号	操作	ロールバック
30	142	バッジ付与	完了 ✅

main.py			
ID	行番号	操作	ロールバック
31	33	データベース移行	完了 ✅
32	78	バッジ初期化	未完了 ✗

一、ストレージエンジン

項目	説明	コード位置
データベースタイプ	PostgreSQL	backend/app/database.py:60
ストレージエンジン	PostgreSQL デフォルト (MVCC)	-
事務支持	<input checked="" type="checkbox"/> ACID 事務	database.py:61 (autocommit=False)
并发制御	MVCC (多バージョン並行制御)	PostgreSQL 自動
特性支持	外部キー、インデックス、JSONB、全文検索	-

二、インデックス

インデックスタイル	インデックス名	テーブル名	フィールド	SQL位置	説明
主キー	users_pkey, posts_pkey, items_pkey	users, posts, items	id	init.sql:3, 28, 75	自動作成
ユニーク	users_email_key, badges_name_key	users, badges	email, name	init.sql:4, 21	UNIQUE 制約
複合ユニーク	likes_key, favorites_key, translations_key	likes, favorites, translations	(post_id, user_id), (post_id, lang)	init.sql:70, 102, 51	重複防止
B-Tree	idx_posts_created, idx_posts_category	posts	created_at DESC, category	init.sql:106, 107	ソート/検索最適化
GIN	idx_posts_tags, idx_posts_attachments	posts	tags, attachments	init.sql:108, migrations/004_*.sql:9	全文/JSON 検索最適化

インデックス総数： 14+ (主キー、ユニーク、複合ユニーク、B-Tree、GIN、外部キー)

五、ロック

ロックタイプ	実装方式	コード位置	説明
行レベルロック	PostgreSQL MVCC 自動	全トランザクション	自動処理並行
表レベルロック	PostgreSQL 自動	DDL 操作	自動処理
ユニーク制約ロック	データベース層	init.sql:70, 102	重複挿入防止
外部キーロック	データベース層	全外部キー 66	参照整合性保 データ量削減
明示的ロック	✗ 未実装	-	アプリ層未実装

グ全文検索

六、関数

関数タイプ	関数名	使用位置	SQLサンプル	説明
組み込み関数	now()	init.sql:15	DEFAULT now()	現在時刻取得
組み込み関数	to_tsvector()	init.sql:108	to_tsvector(...)	全文検索ベクトル化
組み込み関数	coalesce()	init.sql:108	coalesce(tags, '')	NULL値処理
組み込み関数	EXTRACT()	badge_service.py:36	EXTRACT('hour', ...)	時間部分抽出
SQLAlchemy my関数	func.now()	models.py:21	server_default=...	ORM時間関数
SQLAlchemy 関数	func.count()	users.py:90	func.count(...)	カウント関数
SQLAlchemy 関数	func.extract()	badge_service.py:36	func.extract(...)	抽出関数
カスタム関数	✗ 未実装	-	-	PostgreSQL カスタム関数未作成

七、制約

制約タイプ	テーブル名	フィールド/組合せ	SQL位置	説明
主キー	users(id) posts(id) items(id) user_badges(user_id, badge_id)		init.sql:3, 28, 75, 93	PRIMARY KEY
ユニーク	users(email)	badges(name)	init.sql:4, 21	UNIQUE
複合ユニーク	likes(post_id, user_id)	favorites(post_id, user_id) translations(post_id, lang) follows(follower_id, following_id)	init.sql:70, 102, 51, models.py:146	UNIQUE(...)
外部キー	posts(author_id)→users.id	comments(post_id)→posts.id comments(author_id)→users.id likes(post_id)→posts.id likes(user_id)→users.id favorites(post_id)→posts.id favorites(user_id)→users.id user_badges(user_id)→users.id user_badges(badge_id)→badges.id	init.sql:29, 57, 58, 67, 68, 99, 100, 90, 91	ON DELETE SET NULL/CASCADE
非空	posts(title)	users(email, password_hash) posts(title, content) comments(content)	init.sql:4-5, 30-31, 59	NOT NULL
デフォルト値	posts(status)	users(role)→'user', posts(category)→'other', items(status)→'selling'	init.sql:14, 33, 80	DEFAULT

制約総数： 25+ (主キー、ユニーク、外部キー、非空、デフォルト値)

八、複数テーブルクエリ

クエリシナリオ	関連テーブル	クエリ方式	コード位置	SQLサンプル
投稿一覧取得	posts + users	relationship JOIN	posts.py:68	LEFT JOIN users ON...
投稿詳細取得	posts + users + likes + comments	relationship	posts.py:228	自動 JOIN
いいね状態確認	likes	単一テーブルクエリ	posts.py:78-81	SELECT * FROM likes WHERE...
ユーザー統計取得	follows	COUNT 集計	users.py:90-95	SELECT COUNT(*) FROM follows...
ユーザーバッジ取得	user_badges + badges	relationship JOIN	models.py:25	JOIN badges ON...
コメント一覧取得	comments + users	relationship	models.py:67	LEFT JOIN users ON...
メッセージ一覧取得	item_messages + users	手動 JOIN	messages.py:123	JOIN users ON...
通知一覧取得	notifications + users	手動 JOIN	notifications.py:57	JOIN users ON...
バッジチェッククエリ	posts + likes	relationship	badge_service.py:98-99	relationship アクセス
お気に入り一覧取得	favorites + posts	手動クエリ	favorites.py:74-81	SELECT * FROM posts WHERE id IN...

複数テーブルクエリ総数： 10+ 種類

九、ログ（データベースログ）

ログタイプ	実装方式	管理方式	説明
クエリログ	PostgreSQL 自動	Render プラットフォーム 管理	Render PostgreSQL 管理
エラーログ	PostgreSQL 自動	Render プラットフォーム 管理	データベース エラー記録
スローカエリログ	PostgreSQL 自動	Render プラットフォーム 管理	性能監視
トランザクションログ	WAL (Write-Ahead Logging)	PostgreSQL 自動	データ耐久性 保証
アプリケーションアクセス	✗ 不可	Render Dashboard 経由	直接アクセス 不可