

IT研修成果発表

Eチーム:Equipe

1. プロジェクト概要

- テーマ：シンプル雑貨オンライン
- 開発期間：2025/07/01 ~ 2025/07/31
- チーム名：Equipe
- チームメンバー：朝枝教人、鶴喰怜、高木優、坂口あやの
花岡歩、佐藤諒
- 開発目的・ゴール：Spring BootとWeb アプリケーション開発の
基本スキル習得、チーム開発の経験

2-1. 開発したシステムの特徴

- 機能一覧

ユーザー側：

ログイン/ログアウト機能、商品一覧/**カテゴリー別/詳細表示**、カート追加/更新/削除、注文機能、**カート情報保存機能***、**注文情報自動入力機能***、**登録情報確認***、**トークン認証***

(*がついているものは会員登録した場合のみ)

管理者側：

ログイン/ログアウト機能、カテゴリ編集機能、商品情報編集機能、注文情報確認・ステータス更新機能、管理者編集機能

2-2. 開発したシステムの特徴

- アーキテクチャ概要

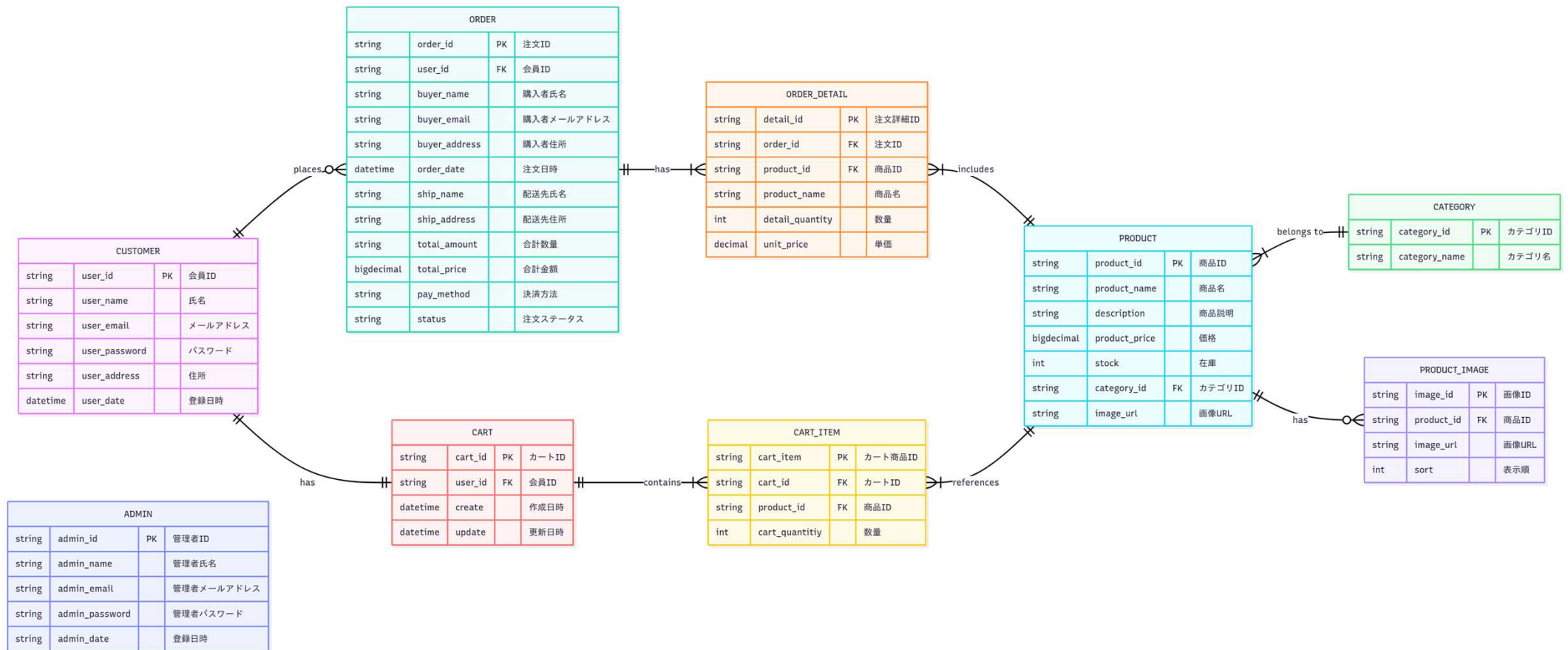
Spring Boot, Java, JPA(Hibernate), H2 Database,
HTML/CSS/JavaScript(Bootstrap), REST API

- 主要な画面フロー

ユーザー側：ログイン→商品一覧ページ→商品詳細→カート→
注文フォーム→注文完了

管理者側：ログイン→ダッシュボード→注文情報確認など

2-3. ER図 (データベース設計)



2-4. API設計（ユーザー）

| APIエンドポイント | | メソッド | 機能 |
|--------------------|--------------------------|--------|-----------------|
| "/api/product" | | GET | 商品一覧取得 |
| | "/{productId}" | GET | 商品詳細取得 |
| | "/category/{categoryId}" | GET | カテゴリ別商品一覧取得 |
| "/api/cart" | | GET | ゲストカート取得（セッション） |
| | | POST | カートに商品追加 |
| | "/items/{itemId}" | PUT | カート内商品の数量変更 |
| | "/items/{itemId}" | DELETE | カート内商品の削除 |
| "/api/order" | | POST | 注文情報登録（ゲスト） |
| "/api/auth" | "/login" | POST | 会員ログイン |
| | "/register" | POST | 新規会員登録 |
| | "/logout" | POST | 会員ログアウト |
| "/api/user" | "/me" | GET | 注文情報入力時の会員情報取得 |
| | "/mypage" | GET | マイページ用会員情報取得 |
| "/api/usercart" | | GET | 会員カート取得（DB） |
| | | POST | カートに商品追加 |
| | "/items/{itemId}" | PUT | カート内商品の数量変更 |
| | "/items/{itemId}" | DELETE | カート内商品の削除 |
| "/api/user/orders" | | POST | 注文情報登録（会員） |

2-4. API設計のポイント（管理者）

| APIエンドポイント | | メソッド | 機能 |
|-------------------------|-----------------|--------|-----------|
| "/api/admin/auth" | "/login" | POST | 管理者ログイン |
| | "/logout" | POST | 管理者ログアウト |
| "/api/admin/categories" | | POST | カテゴリ登録 |
| | "/{categoryId}" | PUT | カテゴリ更新 |
| | | GET | カテゴリ一覧取得 |
| | "/{categoryId}" | DELETE | カテゴリ削除 |
| "/api/admins" | | GET | 管理者一覧取得 |
| | | POST | 管理者登録 |
| | "/{id}" | DELETE | 管理者削除 |
| "/api/admin/order" | | GET | 注文一覧取得 |
| | "/{orderId}" | GET | 注文詳細取得 |
| | "/{orderId}" | PATCH | 注文ステータス更新 |
| "/api/admin/products" | | POST | 商品登録 |
| | | GET | 商品一覧取得 |
| | "/{id}" | PUT | 商品情報更新 |
| | "/{id}" | DELETE | 商品削除 |

3-1. 単体テスト

- テスト項目表をもとにテストコードを作成
- service, controller, repository 単位でロジックを確認
- フレームワークとしてJUnitを使用

AdminAuthControllerTest.javaの例

```
16 class AdminAuthServiceTest {
77     void authenticate_EmptyEmail_ThrowsException() {
78         when(adminRepository.findByAdminEmail(adminEmail:"")).thenReturn(Optional.empty());
79
80         IllegalArgumentException ex = assertThrows(expectedType:IllegalArgumentException.class, () ->
81             | adminAuthService.authenticate(email:"", validPassword));
82         assertEquals(expected:"管理者が見つかりません", ex.getMessage());
83     }
84
85     @Test
86     void authenticate_EmptyPassword_ThrowsException() {
87         when(adminRepository.findByAdminEmail(validEmail)).thenReturn(Optional.of(sampleAdmin));
88
89         IllegalArgumentException ex = assertThrows(expectedType:IllegalArgumentException.class, () ->
90             | adminAuthService.authenticate(validEmail, password:""));
91         assertEquals(expected:"パスワードが正しくありません", ex.getMessage());
92     }
93
94     @Test
95     void authenticate_NullPassword_ShouldThrowException() {
96         when(adminRepository.findByAdminEmail(validEmail)).thenReturn(Optional.of(sampleAdmin));
97
98         IllegalArgumentException ex = assertThrows(expectedType:IllegalArgumentException.class, () ->
99             | adminAuthService.authenticate(validEmail, password:null));
100         assertEquals(expected:"パスワードが正しくありません", ex.getMessage());
101     }
102 }
103
104
105
106
107 }
```


3-2. 結合テスト

- テストケースをまとめたシナリオを作成
- APIとDB、画面の連携を確認
- シナリオごとにmd形式で記述、JSONでテスト結果を返す

| simplezakka結合テストシナリオ一覧 | | | | | |
|------------------------|-----|---------------------|---|---|--------|
| 1. 管理者ログイン機能 | | | | | |
| シナリオ分類 | No. | シナリオ名 | 連携確認ポイント | 期待される主な結果 | 結果(画面) |
| 正常系 | 1-1 | 正しい資格情報でログイン | POST /admin/auth/login→AdminAuthController →AdminAuthService→AdminRepository→DB | セッションにADMIN_SESSIONが格納され、レスポンスが"success"(HTTP 200 OK) | ok |
| | 1-2 | ログイン済み状態からログアウト | POST /admin/auth/logout→AdminAuthController →HttpSession.invalidate() | セッションが破棄され、HTTP 200 OKが返される | ok |
| 異常系 | 1-3 | 登録していないemailでログイン失敗 | POST /admin/auth/login→AdminAuthController →AdminAuthService→AdminRepository | メールアドレス不一致により例外が投げられ、HTTP 401 Unauthorizedと"ログイン失敗"メッセージが返される | ok |
| | 1-4 | パスワードの不一致でログイン失敗 | POST /admin/auth/login→AdminAuthController →AdminAuthService→AdminRepository | パスワード不一致により例外が投げられ、HTTP 401 Unauthorizedと"ログイン失敗"メッセージが返される | ok |
| | 1-5 | フォーム未入力でログイン失敗 | <form>入力チェック→JavaScript→Controllerへ未送信 | 必須項目未入力のため、JavaScriptやブラウザのバリデーションでリクエストがブロックされる | ok |

```
1 ## No. 1-5
2 - テストケース名: ログイン失敗 (異常系 - メールアドレスまたはパスワード未入力)
3
4 - 前提条件:
5 | - ブラウザのHTMLフォームで、メールアドレスまたはパスワードが未入力の状態でログインボタンが押される。
6
7 - 手順:
8 | 1. 管理者ログインフォームに空の値を入れて送信する。
9
10 - 入力データ:
11 | - email: ""
12 | - password: ""
13
14 - 期待結果:
15 | 1. ブラウザの入力バリデーション、またはJavaScript によりリクエスト自体がサーバーへ送信されない。
16 | 2. サーバー側ではコントローラやサービスロジックは実行されない。
17 | 3. フロントエンド側で「入力必須です」という旨のメッセージが表示されること。
```

← → ↺ local:localhost:8080/admin-... ☆ ⓘ ⚙

WELCOME TO "TEINEI"

管理者ログイン画面

E-MAIL

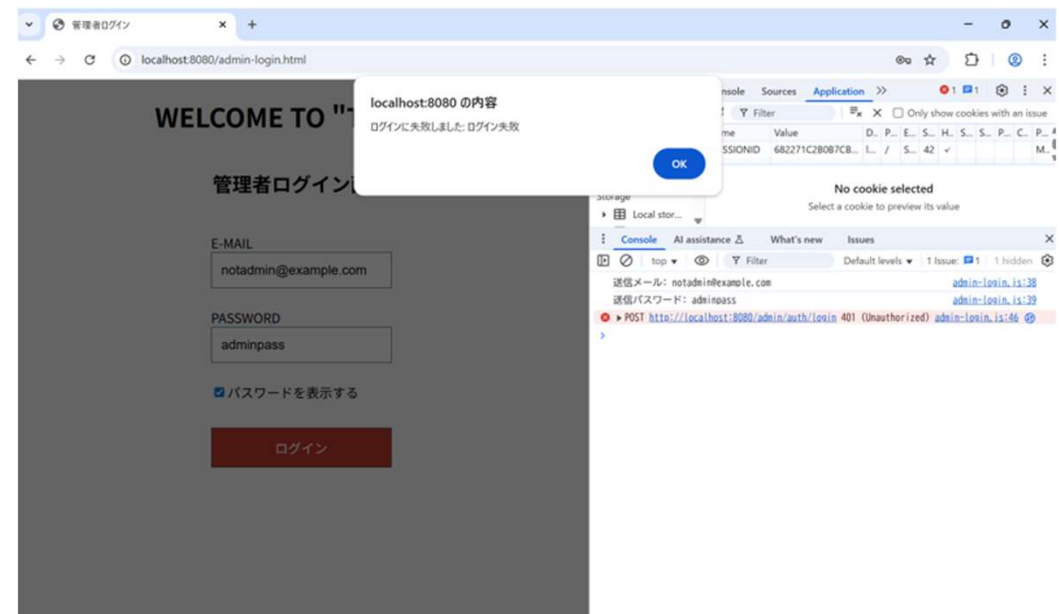
PASS  このフィールドを入力してください。

ログイン

3-3. 総合テスト

- 機能/非機能に分けテストケース一覧を作成
- 実際にブラウザを操作して動作を確認し、画面キャプチャ
- 不具合に対して、開発者ツールによる原因究明
- Excelファイルに総合テスト仕様書としてまとめる

| 1. 管理者ログイン/ログアウト機能 | | | | | | |
|--------------------|-----------|-------|--------------------------|--|---|--|
| No. | テスト対象 | テスト種別 | 目的/観点 | 前提条件 | 操作手順 | 期待結果 |
| ST-AL-001 | 管理者ログイン画面 | 機能 | 有効な管理者アカウントで正常にログインできるか | 管理者 email: admin@example.com pass: adminpass | 1. http://localhost:8080/admin-login.html にアクセス 2. emailに"admin@example.com" 、パスワードに"adminpass"を入力 3. 「ログイン」ボタンをクリック | 1. POST /admin/auth/login が実行され、HTTPステータス200が返ること 2. セッションが開始されること 3. admin-top.html に遷移されること |
| ST-AL-002 | 管理者トップ画面 | 機能 | ログイン済み管理者が正常にログアウトできるか | ST-AL-001 を実行してログイン済みであること | 1. http://localhost:8080/admin-top.html の画面下の「ログアウト」ボタンをクリックする | 1. POST /admin/auth/logout が実行され、HTTPステータス200が返ること 2. セッションが破棄されること 3. admin-login.html にリダイレクトされること |
| ST-AL-003 | 管理者ログイン画面 | 機能 | 非登録のIDでログインしようとした場合の挙動確認 | 管理者アカウント (email: "notadmin@example.com" パスワードに"adminpass"を入力) mは存在しない | 1. ログイン画面にアクセス 2. emailに"notadmin@example.com" パスワードに"adminpass"を入力 3. 「ログイン」ボタンをクリック | 1. POST /admin/auth/login が実行され、HTTP 401 Unauthorized が返ること 2. 画面にエラーメッセージが表示され、ログインできないこと |



3-4. 品質確保の工夫

- 機能ごとにテスト項目を洗い出し、正常系・異常系・境界値・UIイベントの観点に注意し網羅性を意識
- 不具合が見つかったところは即修正 & 再テスト
修正箇所は個人でまとめてテスト完了後にチームに共有する
- テスト結果のエビデンスをテストごとに適した方法で残す
- 実際の利用を想定して発生しうる挙動それぞれに対してテスト
- 開発完了前に動作確認を実施

4-1. ふりかえり：Keep

すり合わせやフィードバックにより、整合性のある設計とシステムを実現
個人としても多くのスキルを習得し、成長を実感できた

チーム

- デザイン性の高いフロントエンド
- 多機能かつDB連携を実現したバックエンドを構築
- サイトを完成させた達成感



個人

- Gitやvs codeの操作に慣れた
- コードを読む力が向上
- 困ったときに考え方も含めて相談できた
- 管理者機能など難所にも挑戦
- 協力や開発の楽しさを実感

4-1. ふりかえり：Problem

計画・設計

- 要件のすり合わせ不足で仕様ズレ
- 見積もり甘さで遅延や偏り発生
- 情報共有・進捗管理不足で連携に支障

技術・チーム運営

- 認証設計に時間がかかった
- コミットやファイル管理が不十分
- チーム内コミュニケーション不足

個人・行動

- 理解不足で指示待ちに
- 得意者依存で成長機会減少

4-1. ふりかえり：Try

計画・設計

- 要件定義で機能範囲を明確化
- 複雑な機能は事前に相談し方針を決定

実装・運用

- ファイル管理やコミットルールを統一
- 進捗報告や質問タイミングをルール化

チームワーク・姿勢

- 事前準備や予習
- 担当外作業にも当事者意識を持つ
- 難しい部分は早めに相談
- 個人とチームの作業バランスを意識

4-2. ふりかえり

■ チームビルディングやプロジェクト管理で工夫した点：

- フェーズごとにリーダーを交代で担当することで、全員が**主体的**に関われるように工夫した
- リーダーは個々の能力に応じて適切な仕事を割り振り、同じ機能を継続して担当してもらうことで引き継ぎを減らし、**理解と開発の効率化**を図った
- 最初に明確に役割を与えることで自分が何をすればいいのかを理解した状態で課題に取り組めた
- **意見を出しやすい雰囲気づくり**を意識し、互いの提案に耳を傾ける文化を育てた

5.システムデモンストレーション

- ① ユーザーホーム画面の紹介
- ② 新規会員登録後ログインユーザーとしての注文完了まで
- ③ 登録した会員情報が保存されていることをログイン画面から確認
- ④ ゲストユーザーによる注文完了まで
- ⑤ 管理者画面から在庫数の減少、完了した注文情報を確認
- ⑥ 管理者画面の紹介

6. 今後の展望・改善提案

■追加したい機能：

- 決済方法追加（カード、ネット決済）
- マイページから会員登録情報編集
- 商品検索
- 注文履歴
- メール自動送信
- 問い合わせ
- お気に入り
- 複数の商品画像
- 管理者削除時にパスワード認証