動的WEB 集中講座（5日間）

動的WEB 集中講座（5日間）

90分×2コマ×5日で鉄則を体に叩き込む

全体設計方針

学習目標

「3つの一致」鉄則を無意識レベルまで習慣化

- Excel設計 → HTML実装 → AI協働 → 動作確認の40分ルーチン完全定着
- 名前統一・事前設計・可視化が自然にできるレベル

反復学習の構造

同じワークフローを毎回繰り返し、難易度だけ上げる

- Day1: 単一フォーム
- Day2: 一覧表示追加
- Day3: 1対多関係
- Day4: 複数関係パターン
- Day5: 自分のサイト統合

Day 1: 基礎の型を体に刻む

1限目（90分）：鉄則の理解と最初の実践

導入（15分）

- **「なぜ3つが一致する必要があるか？」**の実演
- 混乱する例 vs 一致する例をライブデモ
- 鉄則10の配布と説明

実践1：お問い合わせフォーム（60分）

40分ルーチンの完全習得

Step1: Excel設計（5分）

contact_name | contact_email | contact_message | created_at

田中太郎 | tanaka@test.com | 質問があります | 2025-01-15 14:30

佐藤花子 | sato@example.jp | 料金を教えてください | 2025-01-15 15:20

Step2: 手書き図（3分）

[お問い合わせフォーム] → [contacts テーブル] → [管理画面一覧]

contact_name入力 → contact_name保存 → 名前表示

contact_email入力 → contact_email保存 → メール表示

contact_message入力 → contact_message保存 → メッセージ表示

Step3: HTML作成（15分）

- name属性をExcel列名と完全一致
- コメントでDB連携部分をマーキング

Step4: AI指示（10分）

- 統一テンプレートでPHP+SQLite生成依頼

Step5: 動作確認（7分）

- Excel想定データでテスト送信
- データベース内容確認

ふりかえり（15分）

- 40分ルーチンで困った点を全員で共有
- 次回に向けた改善点確認

2限目（90分）：反復練習で定着

実践2：会員登録フォーム（40分）

同じルーチンを別テーマで反復

- user_name, user_email, user_password, user_phone, registered_at
- 40分で完全に完成させる

実践3：アンケートフォーム（40分）

さらに別テーマで反復

- survey_name, survey_age, survey_gender, survey_comment, submitted_at
- 慣れてきたら35分で完成を目指す

1日目まとめ（10分）

- **「40分ルーチンが身についたか？」**セルフチェック
- 明日の予告（一覧表示の追加）

Day 2: 一覧表示で実用性を体感

1限目（90分）：データ表示の追加

復習（10分）

- 昨日のお問い合わせフォームを確認
- 40分ルーチンの再確認

実践4：管理画面作成（70分）

保存したデータを表示する画面

Excel設計拡張（5分）

- 既存データに表示用の想定を追加
- どの項目を一覧で見せるか決定

手書き図更新（3分）

［フォーム］→［DBテーブル］→［管理画面一覧］

↓

［編集］［削除］ボタン

AI指示（15分）

- 一覧表示機能付きのPHP生成
- 編集・削除機能も含める

実装・確認（35分）

- 管理画面の動作確認
- データの編集・削除テスト
- 日本語表示の確認

スタイリング（12分）

- CSSでテーブル表示を見やすく調整

ふりかえり（10分）

2限目（90分）：別テーマでの反復＋応用

実践5：商品管理システム（50分）

`product_name, product_price, product_description, product_stock`

- フォーム＋一覧表示を40分で完成
- 在庫数の数値表示も含める

実践6：イベント管理システム（30分）

`event_title, event_date, event_location, event_capacity`

- 慣れてきたので30分チャレンジ
- 日付型の扱いも経験

2日目まとめ（10分）

- データの「入力→保存→表示→編集」サイクル完全理解
- 明日の予告（関連テーブル）

Day 3：関連テーブルで本格システム

1限目（90分）：1対多の関係理解

概念説明（20分）

- **「なぜテーブルを分けるのか？」**を身近な例で
- カラオケ：歌手→楽曲の関係

- Excel で関係を可視化

実践7：カラオケシステム基礎（60分）

singers テーブル + songs テーブル

Excel設計（8分）

【singers】

singer_id | singer_name

1 | 福山雅治

2 | 安室奈美恵

【songs】

song_id | song_title | singer_id | release_year

1 | 桜坂 | 1 | 2000

2 | CAN YOU CELEBRATE? | 2 | 1997

手書き図（5分）

[歌手登録] → [singers] ← [楽曲登録]

↓ ↓

[歌手一覧] [songs]

↓

[楽曲一覧(歌手名付き)]

AI指示（15分）

- 2テーブル関連システムの生成依頼
- JOIN表示も含める

実装・確認（32分）

- 歌手登録→楽曲登録→関連表示の流れ確認

ふりかえり（10分）

2限目（90分）：関連テーブルの反復練習

実践8：ブログシステム（50分）

categories + articles の1対多

- category_name → article_title, article_content, category_id
- カテゴリ別記事一覧の実装

実践9：予約システム（30分）

services + reservations の1対多

- service_name → reservation_date, customer_name, service_id
- サービス別予約状況の確認

3日目まとめ（10分）

- 1対多関係の設計パターン完全習得確認
- 明日の予告（複数パターンの練習）

Day 4: 多様なパターンで応用力強化

1限目（90分）：ECサイトパターン

実践10：商品-注文システム（70分）

products + orders + order_items の関係

Excel設計（10分）

【products】

product_id | product_name | product_price | product_stock

【orders】

order_id | customer_name | customer_email | order_date | total_amount

【order_items】

item_id | order_id | product_id | quantity | item_price

複雑な関係の可視化（10分）

- 3テーブルの関係を図示
- どのテーブルが「中間テーブル」かを理解

段階的実装（40分）

1. まず商品管理のみ（15分）
2. 注文機能追加（15分）
3. 注文詳細表示（10分）

動作確認（10分）

ふりかえり（20分）

- 複数テーブル設計の難しさと解決策
- Excel事前設計の重要性再確認

2限目（90分）：学習管理システムパターン

実践11：コース-受講生システム（60分）

courses + students + enrollments の多対多関係

- 中間テーブルの役割理解
- 受講状況管理の実装

実践12：図書管理システム（20分）

books + borrowers + loans

- 既存パターンの応用
- 速度重視で基本機能のみ実装

4日目まとめ（10分）

- 様々な関係パターンの理解度確認
- 最終日の準備

Day 5: 自分のサイトに統合&総仕上げ

1限目（90分）：個人プロジェクト統合

各自の既存サイト分析（20分）

- 自分の静的サイトのどこにデータベースが必要か検討
- Excel で追加したい機能を設計

実践13：個人プロジェクト実装（60分）

各自が選んだテーマで実装

- お問い合わせ、予約、商品展示、ブログ等
- 今まで学んだパターンを応用
- 40分ルーチンを意識して実装

進捗共有（10分）

- 隣の人と進捗・困った点を共有

2限目（90分）：発表準備&総まとめ

作品仕上げ（40分）

- 個人プロジェクトの完成
- 動作確認とエラー修正
- 簡単なスタイリング

発表準備（20分）

- **「どんな設計をして、どう実装したか」**を3分で説明準備
- Excel設計→実装→結果の流れで構成

全員発表（25分）

- 1人3分で成果発表
- 使った鉄則、つまずいた点、解決方法を共有

総まとめ（5分）

- **5日間で身についたスキル**の確認
- 今後の学習方針

毎日の共通要素

1日の始まり（各1限目冒頭5分）

- **「今日使う鉄則」**の確認
- 前日の復習クイズ

1日の終わり（各2限目終了前5分）

- **「今日身についた習慣」**の振り返り
- 明日への課題確認

40分ルーチンの時間配分（毎回統一）

1. Excel設計（5分）
2. 手書き図（3分）
3. HTML実装（15分）
4. AI協働（10分）
5. 動作確認（7分）

評価ポイント

プロセス重視（80%）

- **鉄則の習慣化度**：名前統一、事前設計、可視化ができているか
- **40分ルーチンの定着度**：スムーズに作業が流れているか
- **エラー対応力**：つまずいても鉄則に戻って解決できるか

成果物（20%）

- 最終日の個人プロジェクト完成度
- 発表での説明の論理性

配布物リスト

Day 1配布

鉄則10一覧カード（デスクに常備）

40分ルーチンチェックシート
AIへの指示テンプレート集

Day 3配布

関連テーブル設計パターン集
Excel色分けルール

Day 5配布

個人プロジェクト企画書テンプレート
発表用構成シート

成功の判定基準

5日後にできるようになっていること

1. **Excel→HTML→AI→確認の40分ルーチンが無意識にできる**
2. **name属性=フィールド名の統一が当たり前になっている**
3. **関連テーブルの設計をExcelで事前確認する習慣**
4. **エラーが出ても「3つの一致」で原因を見つけられる**
5. **AIへの指示が適切にできる**

DB設計思考の習慣化（新規追加）

6. **Excel設計時に「後で変更しやすいかな？」と自然に考える**
7. **「データが増えても大丈夫？」を意識する習慣**
8. **「誰がアクセスできる？」のセキュリティ観点**
9. **テーブル分割を「面倒」ではなく「必要」と感じる**
10. **「将来こうなったら？」を自然に考える癖**

発言の変化で確認

Before: 「なんでテーブル分けるんですか？」「動けばいいじゃん」 **After:** 「分けておいた方が後で楽ですね」「後で変更する時のことも考えないと」

この変化があれば、DB設計の本質的重要性を理解できた証拠

この10項目ができれば、今後どんなWebアプリケーションでも自力で作れるようになる

PHP/SQLite利用ガイド - Step by Step

PHP/SQLite利用ガイド - Step by Step

このガイドの使い方

HTML/CSSでフォームが作れる人が、AIを使ってバックエンド（PHP+SQLite）を追加するためのガイドです。

Step 1: 企画段階でのデータベース思考

考える3つの質問

1. **「保存したいデータは何？」**
   - 例：名前、メール、メッセージ、投稿日時
2. **「ユーザーが入力するものは？」**
   - 例：名前、メール、メッセージ（投稿日時は自動）
3. **「関連するデータはある？」**
   - 例：お問い合わせ → 返信履歴

企画書への追記項目

【データ項目リスト】

- contact_name（テキスト、必須）

- contact_email（メール、必須）

- contact_message（長文、必須）

- created_at（日時、自動）

Step 2: Figmaデザインでのマーキング

フォーム項目の注記方法

Figmaの各入力欄に **name属性名** を併記

お名前［入力欄］← name="contact_name"

メール［入力欄］← name="contact_email"

メッセージ［テキストエリア］← name="contact_message"

動的コンテンツの識別

- **データベース由来の部分** → 薄青背景で区別
- **繰り返し表示される部分** → 枠線で囲み class="item-list" と注記
- **管理者のみ表示** → 薄赤背景で区別

関連データの表示設計

［商品一覧］

- 商品名 ← products.name

- 価格 ← products.price

- カテゴリ ← categories.name（結合）

Step 3：HTML/CSS段階でのAI協働準備

HTMLマーキング（AIが理解しやすい形）

フォームの準備

```
<!-- DB: contact_form -->

<form method="POST" action="process.php">

    <input type="hidden" name="csrf_token" value="">



    <label for="contact_name">お名前</label>

    <input type="text" name="contact_name" id="contact_name" required>
```

```
    <label for="contact_email">メールアドレス</label>
    <input type="email" name="contact_email" id="contact_email" required>

    <label for="contact_message">メッセージ</label>
    <textarea name="contact_message" id="contact_message" required></textarea>

    <button type="submit">送信</button>
</form>
```

一覧表示の準備

```
<!-- DB: contact_list -->
<div class="contact-list">
    <!-- 繰り返し部分の設計 -->
    <div class="contact-item">
        <h3>[名前]</h3>
        <p>[メッセージの抜粋]</p>
        <time>[投稿日時]</time>
    </div>
</div>
```

Step 4: AIへの指示テンプレート

基本フォーム用テンプレート

以下のHTMLフォームに対応するPHP+SQLiteシステムを作ってください：

【フォーム項目】

- contact_name（テキスト、必須、バリデーション：50文字以内）

- contact_email（メール、必須、バリデーション：メール形式）

- contact_message（長文、必須、バリデーション：500文字以内）

【必要な機能】

- CSRF対策

- バリデーション

- 確認画面

- 送信完了画面

- 管理者向け一覧表示

【使用技術】

- PHP 8.0以上

- SQLite

- セキュリティ対策を含む

初心者でも設置できるよう、設置手順も教えてください。

テーブル関係用テンプレート

以下の関係を持つシステムを作ってください：

【テーブル1: categories（カテゴリ）】

- id（主キー）

- name（カテゴリ名）

【テーブル2: articles（記事）】

- id（主キー）

- title（記事タイトル）

- content（記事内容）

- category_id（外部キー → categories.id）

- created_at（作成日時）

【必要な機能】

- カテゴリ管理（追加・表示・削除）

- 記事管理（追加・表示・編集・削除）

- カテゴリ別記事一覧

- 記事とカテゴリの関連表示

PHP+SQLiteで実装してください。

Step 5: 動作確認の手順

1. ファイル設置確認

process.php が正しく配置されている
データベースファイル（.db）が作成されている
適切な権限設定（書き込み可能）

2. フォーム送信テスト

テストデータで送信成功
バリデーションエラーの確認
CSRF対策の動作確認

3. データ確認

データベースにデータが保存されている
管理画面で一覧表示される
日本語の文字化けがない

4. エラー対応

エラーが出た場合の対処法：

1. エラーメッセージを完全にコピー
2. AIに「このエラーを修正してください：[エラーメッセージ]」
3. 修正されたコードで再テスト

Step 6: よくあるエラーと解決法

データベース接続エラー

エラー例：PDO connection failed

解決法：データベースファイルのパスと権限を確認

文字化け

エラー例：日本語が「???」で表示

解決法：PHPでmb_internal_encoding('UTF-8')を設定

CSRF Token エラー

エラー例：Invalid CSRF token

解決法：セッション開始とトークン生成の確認

Step 7: セキュリティチェックリスト

AIに確認してもらう項目

- SQL Injection対策（プリペアドステートメント）
- XSS対策（htmlspecialchars）
- CSRF対策（トークン）
- バリデーション（サーバーサイド）
- ファイルアクセス権限

チェック用プロンプト

以下のPHPコードのセキュリティをチェックして、

問題があれば修正版を提示してください：

[コードを貼り付け]

特に以下の点を重視してください：

- SQL Injection対策

- XSS対策

- CSRF対策

Step 8: 運用・保守

定期的なメンテナンス

- データベースファイルのバックアップ
- ログファイルの確認
- 古いデータの整理

機能拡張時の考慮点

- 既存データとの互換性
- テーブル構造の変更手順
- データ移行の安全性

付録：プロジェクト構成例

```
project/
├── index.html          # メインページ（静的）
├── contact.html        # お問い合わせフォーム（静的）
├── process.php         # フォーム処理（AI生成）
```

```
├── admin.php           # 管理画面（AI生成）
├── database.db         # SQLiteファイル（自動生成）
├── css/
│   └── style.css       # スタイルシート
└── js/
    └── script.js       # JavaScript（必要時）
```

このガイドを参考に、段階的にバックエンド機能を追加していきましょう。

データベース設計 一致の鉄則

データベース設計 一致の鉄則

実態・技術・脳内イメージを遊離させない10の鉄則

鉄則1：命名の完全一致

HTMLのname属性 = DBのフィールド名 = Excelの列名

☑ 良い例

Excel列名　　：contact_name

HTMLのname　：〈input name=″contact_name″〉

DBフィールド　：contact_name VARCHAR(100)

❌ 悪い例

Excel列名　　：お名前

HTMLのname　：〈input name=″name″〉

DBフィールド　：user_name VARCHAR(100)

なぜ重要？ 頭の中で「あれ？どの名前がどこに対応してる？」と混乱しなくなる

鉄則2：Excelでのデータベース設計確認

必ず実際のデータ例をExcelで作ってから技術実装

手順

1. Excel で実データの例を10行程度作成
2. 列名＝そのままフィールド名として使用
3. データの型・長さを実例から判断

例：お問い合わせシステム

contact_name	contact_email	contact_message	created_at
田中太郎	tanaka@example.com	よろしくお願いします	2025-01-15 14:30
佐藤花子	sato@test.co.jp	料金について教えてください	2025-01-15 15:45

この表をそのままDBテーブルにする

鉄則3：1枚の紙に全体像を書く

脳内イメージ = 手書き図 = 実装

必須要素

[フォーム] → [DBテーブル] → [一覧表示]

↓ ↓ ↓

input name　フィールド名　表示項目

contact_name → contact_name → 名前

contact_email → contact_email → メール

A4用紙1枚に収める（複雑になったら分割する）

鉄則4：日本語→英語変換ルールの統一

変換パターンを決めて一貫適用

基本パターン

- お名前 → contact_name（接頭語＋名詞）
- メールアドレス → contact_email
- お問い合わせ内容 → contact_message
- 投稿日時 → created_at

- 更新日時 → updated_at

関連テーブルの場合

- 商品名 → product_name
- 商品価格 → product_price
- カテゴリ名 → category_name
- 注文日 → order_date

迷ったら「テーブル名_項目名」の形式

鉄則5：データ型の事前決定

Excelでデータ例を見てから型を決める

判断基準

【文字列】

- 短文（50文字以内）→ VARCHAR(100)

- 長文（500文字以内）→ TEXT

- 超長文 → TEXT

【数値】

- 整数（ID、個数）→ INTEGER

- 金額 → INTEGER（円単位で保存）

- 小数点 → REAL

【日時】

- 作成日時 → DATETIME

- 日付のみ → DATE

Excelの実例を見て「この項目なら最大何文字？」を確認

鉄則6：関連の可視化（Excel＋色分け）

複数テーブルの関係もExcelで先に確認

例：商品-カテゴリ関係

categories（薄青背景） ｜ category_id ｜ category_name ｜ |-------------|-------------|｜ 1 ｜ 食品 ｜｜ 2 ｜ 家電 ｜

products（薄黄背景） ｜ product_id ｜ product_name ｜ category_id ｜ product_price ｜ |------------|--------------|-------------|---------------|｜ 1 ｜ りんご ｜ 1 ｜ 150 ｜｜ 2 ｜ 冷蔵庫 ｜ 2 ｜ 50000 ｜

category_id の値が一致することを色で確認

鉄則7：フォーム項目の事前整理

HTMLを書く前に、紙かExcelで項目を整理

テンプレート

項目名（日本語） ｜ name属性 ｜ 必須？ ｜ 型 ｜ 最大長 ｜ 備考

お名前 ｜ contact_name ｜ ○ ｜ 文字列 ｜ 50 ｜

メール ｜ contact_email ｜ ○ ｜ メール ｜ 100 ｜

メッセージ ｜ contact_message ｜ ○ ｜ 長文 ｜ 500 ｜

送信日時 ｜ created_at ｜ - ｜ 日時 ｜ - ｜ 自動

この表通りにHTMLとDBを作る

鉄則8：AIへの指示の統一フォーマット

人間の理解とAIの理解を一致させる

指示テンプレート

以下のExcel表と同じ構造でPHP+SQLiteシステムを作ってください：

【テーブル：contacts】

contact_name VARCHAR(100) NOT NULL

contact_email VARCHAR(100) NOT NULL

contact_message TEXT NOT NULL

created_at DATETIME DEFAULT CURRENT_TIMESTAMP

【HTMLフォームのname属性】

上記フィールド名と完全一致させてください

【必要機能】

- データ保存

- 一覧表示

- セキュリティ対策

Excelイメージ→技術実装が直結

鉄則9：動作確認の3点セット

作ったものが設計通りか必ず確認

確認項目

1. **Excelの想定データがちゃんと入るか？**
2. **HTMLのname属性とDBフィールドが一致しているか？**
3. **一覧表示で期待通りの項目が出るか？**

確認用テストデータ

必ずExcelで作った例と同じデータでテスト

- 日本語を含む
- 最大長ぎりぎり
- 特殊文字を含む

鉄則10：エラー時の対照確認

うまくいかない時は3つの一致を再確認

チェックリスト

HTMLのname属性がDBフィールド名と完全一致？
DBのフィールド型がExcelデータに適合？
日本語英語変換が一貫している？
関連テーブルのIDが正しく参照？
必須項目の設定が一致？

1つずつ確認すれば必ず原因が見つかる

実践例：お問い合わせシステム

Step 1: Excel設計

contact_name	contact_email	contact_message	created_at
田中太郎	tanaka@test.com	質問があります	2025-01-15 14:30

Step 2: HTML（name属性をExcel列名と一致）

```
<input type="text" name="contact_name" required>

<input type="email" name="contact_email" required>

<textarea name="contact_message" required></textarea>
```

Step 3: AIへの指示

上記Excel表と同じフィールド名でPHP+SQLiteを作ってください

Step 4: 確認

- Excel想定データで実際に送信テスト
- DBに想定通りのデータが入っているか確認

まとめ：混乱を避ける心得

1. **名前は全部同じにする**（Excel列名=name属性=DBフィールド名）
2. **実例を先に作る**（Excelで実データ→技術実装）
3. **1枚の紙に全体像**（脳内イメージを可視化）
4. **迷ったら基本ルールに戻る**（テーブル名_項目名）
5. **3点確認でエラー解決**（Excel・HTML・DB の一致確認）

この鉄則を守れば、偏差値に関係なく確実にデータベースシステムが作れます

バックエンド統合における責任分担と学習範囲

バックエンド統合における責任分担と学習範囲

バックエンド統合に必要な5つの要素

全体像

[1] データモデル設計 → [2] DB実装 → [3] バックエンドシステム → [4] フロントエンド連携 → [5] 全体統合

↑人間が責任　↑AI丸投げ　↑AI丸投げ　↑人間が責任　↑人間が責任

責任分担の明確化

🤖 AIに丸投げする領域

[2] DBの実装

- SQLiteファイルの作成
- テーブル作成のSQL文
- インデックスの設定
- データ型の最適化

[3] バックエンド側のシステムの設計・実装

- PHPのビジネスロジック
- CRUD操作の実装
- セキュリティ対策（SQL Injection、XSS等）
- エラーハンドリング
- セッション管理
- バリデーション処理

理由： 複雑で変化が激しく、AIの方が確実で最新の実装ができる

👨‍💼 人間が責任を持つ領域

[1] データモデルの設計

実施方法： AIとExcelを活用した協働設計

人間の責任：

- ビジネス要件の整理
- 必要なデータ項目の特定
- テーブル間の関係性の定義
- 将来の拡張性を考慮した設計判断

AIの活用：

- 設計案の妥当性チェック
- 正規化の提案
- パフォーマンス観点でのアドバイス

ツール： Excel（実データ例作成）+ AI（設計検証）

[4] フロントエンド側のコード修正によるDB連携の実装

実施方法： AIに助けられながら手書きコーディング

人間の責任：

- HTMLのフォーム要素とDB項目の対応付け
- フロントエンドとバックエンドの連携ポイント特定
- ユーザー体験を考慮した画面遷移設計
- エラー時の画面表示制御

AIの活用：

- コード生成の支援
- デバッグの支援
- 最適化の提案

実装内容：

```
<!-- 人間が責任を持つ部分 -->

<form method="POST" action="process.php">
```

```
    <input type="text" name="contact_name" required>

    <!-- ↑ name属性とDB項目の対応を人間が設計 -->



    <!-- AI生成の部分を統合 -->

    <?php include 'form_handler.php'; ?>

</form>
```

[5] 全体の統合

実施方法： 従来ワークフローの拡張 + 10の鉄則徹底

人間の責任：

- 企画段階でのデータ要件定義
- Figmaデザインでのデータ流れ設計
- コーディング時の統合ポイント管理
- 動作確認と品質保証

従来ワークフローの拡張

従来： 企画 → Figmaデザインカンプ → HTML/CSSコーディング

新版： 企画 + データ設計 → Figmaデザインカンプ + DB連携設計 → HTML/CSS + バックエンド統合

各段階での具体的作業

1. 企画段階の拡張

従来の企画内容に追加：

- **データ項目リスト作成**（Excelで実例作成）

- **ユーザーアクション整理**（何をしたら何が保存される？）
- **権限設計**（誰が何を見られる？）

成果物：

【企画書】

サイト目的：お問い合わせ受付

ターゲット：既存顧客

主要機能：問い合わせフォーム + 管理画面

【データ設計（Excel）】

contact_name | contact_email | contact_message | created_at

田中太郎 | tanaka@test.com | 質問があります | 2025-01-15 14:30

【権限設計】

一般ユーザー：フォーム送信のみ

管理者：一覧表示、返信、削除

2. Figmaデザイン段階の拡張

従来のデザインに追加：

- **フォーム項目へのname属性併記**
- **動的コンテンツ部分の識別**（背景色で区別）
- **データフローの可視化**

成果物：

[Figmaデザイン]

お名前［入力欄］← name="contact_name"

メール［入力欄］← name="contact_email"

メッセージ［テキストエリア］← name="contact_message"

[データフロー図]

フォーム入力 → DB保存 → 管理画面表示 → 返信作成

3. コーディング段階の拡張

従来のHTML/CSSに追加：

- **10の鉄則適用**（命名統一、事前設計等）
- **AI生成コードとの統合ポイント設計**
- **動作確認の体系化**

作業フロー：

1. HTMLコーディング（name属性はDB設計通り）
2. AIにバックエンド生成依頼
3. 生成されたPHPファイルとHTMLの統合
4. 動作確認（Excel想定データでテスト）
5. エラー対応（鉄則に基づく原因特定）

10の鉄則の徹底適用

統合時に特に重要な鉄則

鉄則1：命名の完全一致

企画書のデータ項目 = Figmaの注記 = HTMLのname属性 = DBフィールド名

鉄則2：Excelでの事前設計

統合前に必ずExcelで全データの流れを確認

鉄則3：1枚の紙に全体像

企画 → デザイン → フロント → バック → 統合の全体を可視化

鉄則6：関連の可視化

複数画面、複数テーブルの関係をExcel色分けで明確化

鉄則9：動作確認の3点セット

1. Excelの想定データが正しく流れるか？

2. フロントとバックの連携ポイントは正しいか？

3. 全体のユーザー体験は期待通りか？

学習における重点配分

重点的に学習する領域（60%）

- データモデル設計思考
- フロントエンド連携技術
- 全体統合のプロジェクト管理

理解するが深入りしない領域（30%）

- バックエンドの仕組み概要
- セキュリティ対策の基本知識
- データベースの基本概念

完全にAI任せの領域（10%）

- PHPの詳細文法
- SQLの複雑な構文
- サーバー設定・運用

この分担の利点

学習効率の最大化

- 重要な設計思考に集中できる
- 技術的な詰まりでの学習停止を回避
- 実用的なシステムを短期間で完成

実務との適合性

- 企業でもAIツール活用が主流になる
- 設計力と統合力が差別化ポイント
- チーム開発での役割分担に近い

将来への拡張性

- 技術スタックが変わっても応用可能
- AI技術の進歩に合わせて活用範囲拡大
- 本質的なスキル（設計・統合）は普遍的

成果として身につく能力

データモデル設計力

- ビジネス要件をデータ構造に落とし込める
- 将来の拡張性を考慮した設計ができる
- Excel + AIでの効率的な設計プロセス習得

統合プロジェクト管理力

- 複数の技術要素を組み合わせて完成品を作れる
- 問題発生時の原因特定と解決ができる
- 品質保証の観点でシステム全体を評価できる

AI協働力

- 適切なタスク分担でAIを活用できる
- AI生成物の品質評価と改善指示ができる
- 人間の責任範囲を明確に理解している

これらの能力は、技術が変化しても通用する普遍的なスキル