JBS 中級コース(案1)／Day2 オリエンテーション／2026-05-21

Day2 事前準備

Day2・Day3 は、新しいリポジトリを Day1 と同様の手法で Fork して進めます。本書は事前セットアップの手順に加え、上流工程×AI のいま、設計書を読む観点、参照すべき情報源、一般的な基準までまとめた一冊です。生成待ちの時間に読めば、検証の質が上がります。

SECTION 01

はじめに

Day2 は Day1 の設計書を実装に落とす日です。全部を実装しきる日ではありません。1 本を深く実装し、生成物を自分で検証して判定できる状態をめざします。

Day1 の成果物

ファイル	内容
`docs/requirements.md`	要件定義書。MoSCoW 分類、SLA 定義、状態遷移規則、要確認事項の解消経緯
`docs/data_model.md`	ER 図、状態遷移図、カラム定義、インデックス方針
`docs/openapi.yaml`	API 仕様、エラーコード一覧、認可仕様
`docs/sequence_diagrams.md`	起票・状態遷移・SLA 違反の3シーケンス図、層責務の分担表

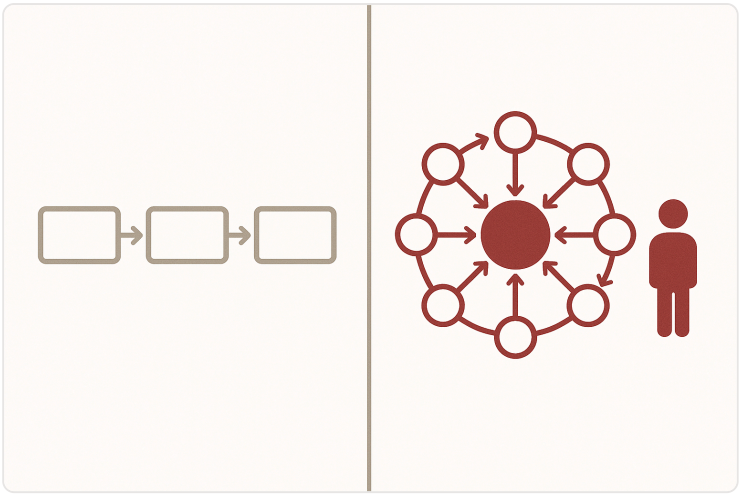
Day2 でめざす状態

- ☑ チケット起票 API を設計書から3層で実装し、なぜこの実装かを説明できる
- ☑ 状態遷移と SLA は、Claude の生成物を設計書と照合して受け入れ可否を判定する
- ☑ 保証内容を説明できるテストが手元にある

SECTION 02

上流工程×AI の現在地

この研修の検証サイクルが、なぜ「いまのトレンドの本流」なのかを先に共有します。



左：従来＝人が工程を直列で進める ／ 右：AI 駆動＝エージェントが全工程を協調し、人は検証・判定の位置に立つ

- **従来（左）**：要件定義 → 設計 → 実装 → テストを人が順に手で進める
- **AI 駆動（右）**：AI エージェントが複数工程を協調して進め、人は「検証・判定・説明責任」を担う位置に移る

いま起きている変化

- ☑ 「AI がコードを書く」から「**AI が開発の仕事を進める（Agentic SDLC）**」への移行が本命トレンド
- ☑ **仕様駆動開発（Spec-Driven Development）**の浸透。仕様を文書で定め、それを正に AI へコード生成させる進め方が広がっている
- ☑ 富士通：LLM「Takane」＋AI エージェントで要件定義〜結合テストを自動化し、3 人月の改修を 4 時間に短縮した実証を公表
- ☑ 電通総研：上流工程の半自動化で、要件定義・基本設計に 30% の生産性向上を確認し本格運用を開始
- ☑ 結論：AI 出力を評価し、設計に照らして判定できる**検証・要件定義スキル**の価値が上がっている

Day2 の検証サイクル（設計書を正に AI 生成物を照合して判定）は、この「仕様駆動 × Agentic SDLC」の本流です。速く作ることより、出力を設計に照らして判定できる力が、いま市場で評価されます。出典は末尾の参照一覧に記載しています。

SECTION 03

事前セットアップ

Day2・Day3 は、第1回とは別の新しいリポジトリで進めます。受講者全員が Day1 完成版の設計書から再開します。git コマンドの入力や、ファイルの上書きはありません。第1回のリポジトリは別リポジトリとしてそのまま残ります。

手順（Fork とクローン）

1. 事務局または講師から共有される Day2・Day3 用リポジトリの URL を、ブラウザで開く
2. 画面右上の「Fork」→「Create fork」をクリックし、自分のアカウントにコピーを作る
3. Fork した自分のリポジトリを、Day1 と同じ手法で VSCode に取り込む（クローン）

完了すると、Day1 完成版の設計書（`docs/` 配下）が入った自分用リポジトリが手元に用意できます。

うまくいかないときの確認

状況	確認すること
リポジトリの URL が分からない	事務局または講師に確認してください
「Fork」ボタンが見つからない	リポジトリ画面の右上、Star ボタンの近くにあります
Fork できたか分からない	画面左上のリポジトリ名が「自分のアカウント名／…」になっていれば成功です
VSCode への取り込み方が分からない	Day1 の事前セットアップと同じ手順です

Claude に相談する

表で解決しないときは、この資料の PDF を Claude Code に添付し、次のように送信してください。今の状況に合わせた操作を案内します。

CLAUDE へのメッセージ例

```
Day2 の事前セットアップでつまずきました。添付資料の手順を実行中です。
今やろうとしている操作:（例: Fork したリポジトリの VSCode への取り込み）
画面の状態:（表示されている内容やメッセージをそのまま書く）
どう操作すればよいか教えてください。
```

解決しない場合

Claude の案内でも解決しない場合は、無理に進めず、その旨を控えておいてください。Day2 開講直後の冒頭セッションで、講師が画面を共有しながら一緒に対応します。事前に完了していなくても受講には支障ありません。

当日の流れ

- [x] 冒頭セッションで、Fork とクローンができているかを講師が画面共有しながら一緒に確認
- [x] 全員が Day1 完成版の同じ起点に揃ったことを確認してから本編へ
- [x] 以降、設計書をもとに実装と検証を進行

SECTION 04

アンケートへの回答

Day1 でいただいた声に対して、Day2 で何を変えたかを共有します。

Day1 でいただいた声	Day2 での対応
生成物が多く、検証しきれない	1 ブロックごとに題材専用チェックリストで判定する「検証サイクル」を導入。生成量も削減
出力の妥当性を確認できず、説明責任が持てない	生成前に正解条件と設計書の根拠を言語化。Session 07 で説明ドリル
進行が速く、理解が追いつかない	全エンドポイント実装をやめ、1 本を深く。残りはレビュー主体に変更
トークンを想定以上に消費した	3 層を 1 プロンプトでまとめず、1 ファイル単位で依頼。Session 07 で実測を振り返り
用語が多く、Claude に不慣れ	用語が初出する箇所に最小 Tips を挿入。本書末尾にも用語の早見表
ターミナル操作に迷った	Day2・Day3 は Fork 方式に変更し、git コマンド入力をなくした

SECTION 05

Day2 の進め方：検証サイクル

「プロンプトを打つ → 出力を Apply する → 次へ」は繰り返しません。1 ブロックごとに次の 4 ステップを回します。

STEP 1

言語化

生成前に「何を満たせば正解か」を1〜2行書く。設計書のどこが根拠かを指す。

STEP 2

生成

範囲を絞って Claude に作らせる。1プロンプトで3層をまとめない。

STEP 3

検証

題材専用チェックリストで生成物を1項目ずつ確認する。

STEP 4

判定

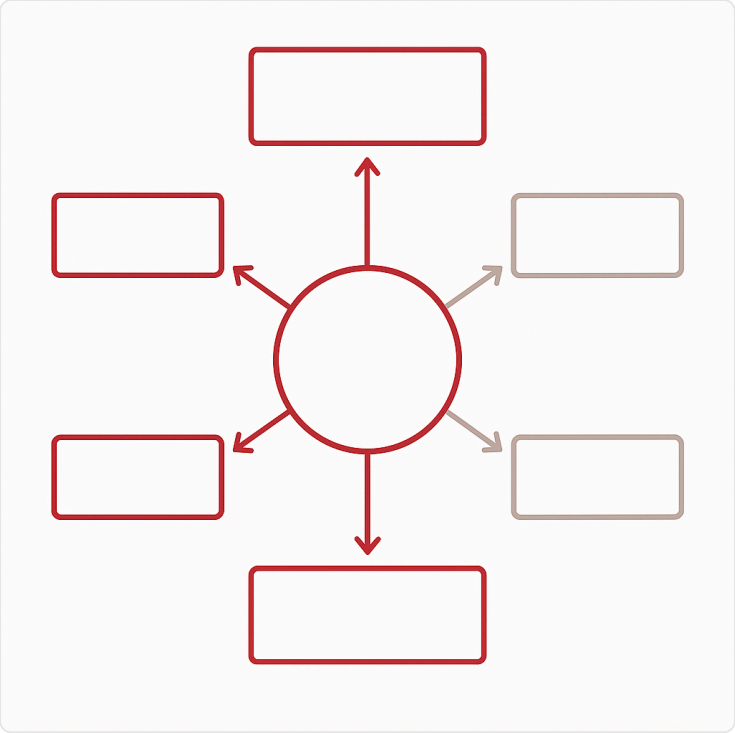
受け入れか差し戻しかを自分で決める。差し戻すなら理由を1行書く。

> **Tips：このサイクルの狙い**
>
> 「レビューしきれない」は、見る観点が言語化されていない状態です。STEP 1 で正解条件を書けば STEP 3 で見る箇所が決まります。「説明責任が持てない」は、根拠の出どころを追っていない状態です。STEP 1 で設計書のどこ由来かを指せば、根拠を示せます。

SECTION 06

設計書を読む観点

生成物を判定するには、設計書を「どの観点で読むか」が決まっている必要があります。本研修で使う 5 観点です。



中心＝検証対象、5 方向＝レビューの観点

- **要件トレーサビリティ**：要件 → 設計 → 実装 → テストが追跡できるか。`requirements.md` の項目が実装に反映されているか
- **データモデル整合**：`data_model.md` のカラム・型・PK/FK と実装が一致するか。勝手な列の追加がないか
- **API 設計**：`openapi.yaml` のパス・スキーマ・エラーコードと一致するか。REST の原則（リソース指向、適切なステータスコード）に沿うか
- **層責務**：router → service → repository の単方向依存。検証や SQL が router に漏れていないか
- **テスト境界値**：「ちょうど」と「1つ外」の両方を突いているか。何を保証するテストか説明できるか

非機能要件は思いつきでなく **ISO/IEC 25010 のシステム/ソフトウェア品質特性**（機能適合性・性能効率性・互換性・使用性・信頼性・セキュリティ・保守性・移植性）の枠で抜けを確認すると、観点の漏れが減ります。

SECTION 07

参照すべき情報源（優先順位）

判定に迷ったとき、どの情報を「正」とするか。優先順位を固定しておくとぶれません。

優先	情報源	使いどころ
1	`docs/` の設計書	この題材の正。要件・データ・API・シーケンスはここを最優先
2	`CLAUDE.md`	技術スタック、ディレクトリ構成、3層の依存方向、禁止事項
3	公式ドキュメント	Claude Code / FastAPI / SQLAlchemy / pytest の正しい書き方
4	標準仕様	OpenAPI 仕様、REST の一般原則、ISO/IEC 25010 の品質特性

本編と補足で値が食い違うとき

ヒアリング本編と補足・制約書で値が違う箇所があります（レスポンス「3 秒」対「1 秒」など）。**後出しの確定値（補足・`architecture_constraints.md`）が正**で、解消経緯は `requirements.md` 第8節にあります。気づけたら Day1 で扱った「情報源の突き合わせ」の成果です。

SECTION 08

一般的な基準と Day2 の前提

SLA だけでなく、判定に使う一般的な基準をまとめます。生成待ちにここを確認しておくと検証が速くなります。

SLA の要点

優先度	初回応答	解決
high	30 分以内	4 時間以内
medium	4 時間以内	1 営業日以内
low	1 営業日以内	3 営業日以内

営業時間は平日 9:00-19:00。営業時間外と土日祝は経過時間に数えません。全優先度で同じ閾値なら、公開デモと同じ誤実装です。

テスト・CI の一般基準

- カバレッジは目的でなく結果。数字あわせより「何を保証するか説明できるテスト」を優先（本研修では未達でも CI を fail させない設定）
- CI 品質ゲートの基本：lint（書式・規約）／型チェック／テスト／ビルド成功。1 つでも赤なら次へ進めない
- コードレビューの基本観点：単一責任、単方向依存、入力検証の位置、例外・エラー形式の一貫、命名の一貫

状態遷移の要点

- 許可：open → in_progress → waiting_customer → resolved → closed、waiting_customer ⇄ in_progress、resolved → open（再オープン）
- 禁止：closed → open（最終状態）、open → resolved の飛び越し
- 遷移時は AuditLog に before / after を記録

SECTION 09

Claude Code とモデルの最新

研修で使うツールの現在地です（2026 年 5 月時点。出典は末尾）。

- 最新の一般提供モデルは **Claude Opus 4.7**（2026 年 4 月。複雑な推論とエージェント的コーディング向け。画像解像度が約 3 倍に向上）
- Code with Claude 2026（SF／ロンドン／東京）は新モデルでなく **エージェント基盤の強化**に注力。マルチエージェントセッション等が公開ベータ
- Claude Code：プラグインの zip／URL 取得対応、Agent SDK のフラグ拡充、Fast モードが Opus 4.7 既定、サブエージェント進捗のキャッシュ化でトークンコストを約 1/3 に削減

> **研修への含意**
>
> モデルとツールは「コードを書く」から「開発を進めるエージェント基盤」へ重心が移っています。だからこそ、出力を設計に照らして判定し説明できる人の役割が、相対的に重くなります。Day2・Day3 はその訓練です。

SECTION 10

Tips と注意

Tips

> **Tips：トークン節約**
>
> 3 層を 1 プロンプトで生成させると、Claude が app/ 全体を読み直して差分が長くなり、読み返せないまま消費が増えます。範囲を 1 ファイルに絞ると、読む量・出る量・修正回数が減ります。

> **Tips：レビューの型**
>
> 生成前に「正解条件」を 1 行書くと、レビューで見る箇所が決まります。観点を先に言語化していないと、出力に引きずられて「それらしいので OK」と流してしまいます。

> **Tips：説明責任の持ち方**
>
> 「AI が出したから正しい」ではなく「要件に照らして自分が判定したから正しい」と言える状態をめざします。根拠を `docs/` のどこに書いてあるかで示せれば、上司にも説明できます。

注意

> **公開デモは「直すべき題材」**
>
> Day1 で観察した公開デモは、title 未検証・状態遷移無検証・SLA 一律1時間という改善余地を意図的に含んでいます。Claude が同じ誤りをしないか、検証で確かめます。

Apply して次へ、を繰り返すと検証できる量を超えます。1 ブロックごとに STEP 3・STEP 4 を必ず通してください。Day2・Day3 は Fork 方式のため、git コマンドやファイル上書きの操作はありません。

SECTION 11

用語早見表

手が止まったとき、待ち時間に目を通しておくと迷いが減ります。

用語	これだけ押さえる
Swagger UI	実装した API をブラウザから手で叩く画面（`/docs`）。Try it out → 入力 → Execute で結果を確認
SLA	対応の時間目標。優先度ごとに初回応答・解決の時間が決まり、超過を違反として検出
ruff	コードの書式・規約チェッカー。CI で自動実行、違反で赤
mypy	型の整合性チェッカー。CI で自動実行、違反で赤
uvicorn	FastAPI アプリを起動するサーバー。`uvicorn app.main:app --reload`
pytest	テスト実行ツール。`pytest tests/ -q`
Agentic SDLC	AI が複数の開発工程をまたいで仕事を進める考え方。人は検証・判定を担う
3層	router=HTTP受け口、service=業務ロジック、repository=DB操作。依存は router→service→repository の一方向

SECTION 12

参照・出典と、持ち帰る視点

設計書・指示書

場所	何を見るか
`docs/requirements.md`	要件、SLA 定義、状態遷移規則、要確認の解消経緯（第8節）
`docs/data_model.md`	カラム・型・PK/FK・index
`docs/openapi.yaml`	エンドポイント、スキーマ、エラーコード
`docs/sequence_diagrams.md`	3 ユースケースの流れ、層責務
`docs/architecture_constraints.md`	非機能要件（レスポンス、可用性、保持期間、イメージサイズ）
`CLAUDE.md`	技術スタック、構成、3層の依存方向、禁止事項

公式ドキュメント・最新動向の出典

- Claude Code 公式：https://code.claude.com/docs/
- Claude Code 最新更新（2026 W19）：https://code.claude.com/docs/en/whats-new/2026-w19
- Anthropic News：https://www.anthropic.com/news
- Claude Platform リリースノート：https://platform.claude.com/docs/en/release-notes/overview
- FastAPI：https://fastapi.tiangolo.com/
- SQLAlchemy 2.0：https://docs.sqlalchemy.org/en/20/ ／ pytest：https://docs.pytest.org/
- 富士通 AI ドリブン開発（2026/02/17）：https://global.fujitsu/ja-jp/pr/news/2026/02/17-01
- 電通総研 上流工程半自動化：https://www.dentsusoken.com/news/release/2025/1009.html
- 仕様駆動開発（SMFG DX-link）：https://www.smfg.co.jp/dx_link/article/0209.html
- AI駆動開発 2026（Qiita）：https://qiita.com/k-yoshinobu/items/d75f8926e8aab47ac232

持ち帰る3点

- 生成前に正解条件を1行書くと、レビューで見る箇所が決まる

- [x] 範囲を絞った生成は、検証できる量に収まり、トークンも減る
- [x] 「AIが出したから」ではなく「要件に照らして自分が判定したから」と言えれば説明責任を持てる

Day2 の到達点

1 本の API を設計書から実装し、生成コードを設計書と照合し、実装の根拠を説明できる状態。残りの実装は Claude に任せ、レビューに回ります。この力が、いまの上流工程で最も問われています。

制作 Givery／JBS 中級コース 案1 Day2 オリエンテーション／2026-05-21