0001 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：基礎

24時間稼働の受注システムで、障害時の停止時間を最小化したい。一方、導入費用には上限がある。方式選定で最初に行うべきこととして最も適切なものはどれか。

ア．可用性目標と許容停止時間を明確にし、必要な冗長度と費用を比較する。
イ．単一構成を前提にし、障害時は運用手順だけで復旧する。
ウ．性能要件だけで方式を決め、可用性は実装後に検討する。
エ．最も高価な冗長化方式を採用してから費用を調整する。

答え・解説 正答：ア

ア：方式設計では要求を定量化し、可用性とコストなどのトレードオフを比較して決める。
イ：高可用性が要求される場合、単一障害点を残す構成は要求未達となり得る。
ウ：可用性は方式選定に強く影響するため、実装後に回すのは遅い。
エ：費用上限との整合を確認せず方式を先に固定すると過剰設計になり得る。
総合解説：システム方式は、機能だけでなく可用性、性能、コスト、運用性など複数要求のバランスで決める。特に非機能要求は数値や条件を明確にして比較可能にする。

対象：2026年度 / 基準日 2026-09-01

0002 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：標準

アクセス量が季節によって10倍程度変動し、処理は複数サーバへ分散可能なWebサービスで、需要増に追随しやすい方式はどれか。

ア．アクセス増加時もサーバ台数を変えずキュー長だけを増やす。
イ．複数インスタンスを追加・削減できる水平スケーリングを採用する。
ウ．全処理を一つの巨大トランザクションにまとめる。
エ．単一サーバのCPUを最大構成に固定する。

答え・解説 正答：イ

ア：台数を固定するとピーク時の容量不足を解消できない。
イ：分散可能な処理では水平スケーリングにより需要に応じた容量調整がしやすい。
ウ：トランザクション巨大化はスケーラビリティを悪化させる可能性がある。
エ：垂直拡張には上限があり、需要変動への細かな追随には向きにくい。
総合解説：負荷特性と処理の分割可能性を踏まえてスケーリング方式を選ぶ。変動が大きく並列化可能なら、台数で調整できる方式が有力である。

対象：2026年度 / 基準日 2026-09-01

0003 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：基礎

疎結合なアーキテクチャを採用する主な狙いとして最も適切なものはどれか。
ア．データ整合性を常に単一DBの直列実行だけで保証する。
イ．インタフェース仕様を不要にする。
ウ．構成要素間の依存を減らし、変更影響を局所化しやすくする。
エ．全ての処理を必ず同一プロセス内で実行する。

答え・解説 正答：ウ

ア：疎結合は単一DBを必須とする概念ではない。
イ：疎結合でも明確な契約・インタフェースは必要である。
ウ：疎結合化は構成要素間の依存を抑え、独立変更や置換をしやすくする。
エ：同一プロセスへの集約は結合度を高める方向である。
総合解説：疎結合は変更容易性や独立性を高める一方、分散化による通信障害、整合性、監視など別の複雑さを伴う。

対象：2026年度 / 基準日 2026-09-01

0004 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：応用

注文受付後、配送会社への通知に数秒の遅延は許容されるが、配送会社の一時停止で注文受付まで失敗させたくない。適切な連携方式はどれか。
ア．配送会社停止中は注文機能全体を停止する。
イ．注文DBのテーブルを配送会社に直接更新させる。
ウ．注文処理の同一トランザクション内で配送会社APIの応答を必須にする。
エ．配送通知をメッセージキューに登録し、配送連携処理が後で取り出す。

答え・解説 正答：エ

ア：不要なサービス停止となり、障害の波及を抑えられない。
イ：DB直接共有は境界を曖昧にし、結合度と整合性リスクを高める。
ウ：外部障害が注文受付へ直結し、可用性要件に反する。
エ：非同期化により受付処理と外部通知を分離でき、一時障害を吸収しやすい。
総合解説：遅延許容と障害分離が重視される場合、非同期メッセージングは有効である。ただし重複配送、順序、再試行、監視を設計する必要がある。

対象：2026年度 / 基準日 2026-09-01

0005 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：標準

参照回数が多く更新頻度が低い商品マスタの応答性能を改善するためキャッシュを導入する。最も重要な設計上の検討はどれか。

ア．キャッシュの更新・失効条件と許容できるデータ鮮度を定める。
イ．キャッシュ導入後は元DBを削除する。
ウ．全データを無期限に保持し、更新通知を無視する。
エ．キャッシュとDBの値が異なっても常にキャッシュを正とする。

答え・解説 正答：ア

ア：性能向上と鮮度・整合性のバランスを決める中心論点である。
イ：永続データの正本を安易に削除する設計ではない。
ウ：無期限保持では陳腐化した値を返す危険がある。
エ：正本と整合しない値を無条件で優先するのは不適切である。
総合解説：キャッシュは応答性能を高めるが、失効、更新伝播、障害復旧時の整合性などを設計しなければならない。

対象：2026年度 / 基準日 2026-09-01

0006 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：標準

Webサーバは2台に冗長化したが、全通信が1台のロードバランサを必ず通る構成である。この構成の評価として適切なものはどれか。

ア．Webサーバが2台なので単一障害点は存在しない。
イ．ロードバランサが停止すると全体へ影響するため、可用性目標に応じて冗長化を検討する。
ウ．ロードバランサはネットワーク機器なので障害しない。
エ．Webサーバを3台に増やせばロードバランサ障害も解消する。

答え・解説 正答：イ

ア：共通経路に単一装置があるため単一障害点が残っている。
イ：共有部品の故障影響を評価し、必要なら冗長化するのが適切である。
ウ：機器種別にかかわらず故障可能性はある。
エ：Webサーバ台数を増やしても共通ロードバランサの単一障害点は残る。
総合解説：冗長化は一部だけでなく、要求されたサービス経路全体で単一障害点を洗い出す必要がある。

対象：2026年度 / 基準日 2026-09-01

0007 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：標準

小規模な新規システムで、チームは5人、機能数は少なく、短期リリースが最優先である。最初から多数のマイクロサービスへ細分化する判断について最も適切なものはどれか。

ア．チーム人数とアーキテクチャ選択は一切関係しない。
イ．マイクロサービスは常に最適なので細分化する。
ウ．分散運用の複雑性も含め、変更独立性などの便益が上回るか評価して決める。
エ．モノリスではAPIを作れないので必ず分割する。

答え・解説 正答：ウ

ア：運用能力やチーム構成は方式の実現可能性に影響する。
イ：方式は要求や組織条件に依存し、常に一択ではない。
ウ：分散化は独立デプロイ等の利点がある一方、監視・通信・整合性などのコストが増える。
エ：モノリスでも内部・外部APIは設計可能である。
総合解説：アーキテクチャは流行ではなく、機能規模、変更頻度、チーム、運用成熟度、非機能要求を踏まえて選ぶ。

対象：2026年度 / 基準日 2026-09-01

0008 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：応用

複数地域に分散した参照サービスで、ネットワーク分断時も各地域で参照を継続することを最優先し、数秒のデータ反映遅延は許容される。適切な設計方針はどれか。

ア．複製は行わず各地域が独立に正本を更新する。
イ．全参照を必ず単一地域のDBに同期接続させる。
ウ．分断時は全地域の参照を停止して完全一致だけを優先する。
エ．各地域へ複製し、許容された範囲で結果整合性を採用する。

答え・解説 正答：エ

ア：複数正本を無制御にすると競合や不整合が生じる。
イ：単一地域依存は分断時の可用性を下げる。
ウ：要求では継続利用が最優先なので停止は不適切である。
エ：許容遅延があるなら複製と結果整合性で可用性を優先できる。
総合解説：分散システムでは一貫性、可用性、分断耐性などの制約を踏まえ、業務が許容する鮮度や競合処理まで含めて方式を決める。

対象：2026年度 / 基準日 2026-09-01

0009 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：基礎

1日に一度、前日分の大量ログを集計して翌朝までにレポートを作る処理に適した方式はどれか。
ア．一定単位でまとめて処理するバッチ処理
イ．利用者端末だけで完結するピアツーピア処理
ウ．全件を一件ずつ同期APIで外部へ送る方式
エ．イベント発生ごとに必ず人手承認する対話処理

答え・解説 正答：ア

ア：締切までにまとめて大量処理する要件にはバッチ方式が適している。
イ：端末分散処理を選ぶ根拠がない。
ウ：外部同期呼出しを大量に行う必然性がなく、障害影響も大きい。
エ：大量定型処理の要件に対して人手介在は非効率である。
総合解説：業務の即時性、処理量、締切、相互作用の必要性からオンライン・バッチなどの処理方式を選ぶ。

対象：2026年度 / 基準日 2026-09-01

0010 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：標準

既存基幹システムとの連携で、相手側が夜間に固定長ファイルだけを受け付ける。この条件を踏まえた設計として適切なものはどれか。
ア．相手側仕様を無視してリアルタイムREST APIだけを採用する。
イ．連携制約を方式設計の入力とし、必要なら内部のリアルタイム処理と外部ファイル連携を分離する。
ウ．固定長ファイルという理由だけで新システム全体もバッチ専用にする。
エ．連携要件を実装担当者へ伝えず実装時に判断させる。

答え・解説 正答：イ

ア：相手側制約を満たせず連携できない。
イ：外部制約を満たしつつ内部方式を必要以上に縛らない設計が適切である。
ウ：局所的な制約を全体へ拡大する必要はない。
エ：重要な制約は方式設計段階で明示しなければならない。
総合解説：方式設計では外部システム、既存資産、法規、運用時間などの制約を入力として扱い、影響範囲を局所化する。

対象：2026年度 / 基準日 2026-09-01

0011 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：応用

ピーク時に毎秒600件の要求があり、1台で毎秒250件を安定処理できるサーバを用いる。1台故障してもピークを処理したい。少なくとも何台必要か。

ア．最小性を考えず余裕を大きく取り、6台必要とする。
イ．通常時3台で足りる点だけを見て、故障後500件/秒になることを無視する。
ウ．4台構成とし、1台故障後も3台×250=750件/秒を確保する。
エ．5台なら要件を満たすが、必要最小台数という条件を満たさない。

答え・解説 正答：ウ

ア：6台は要件に対して過剰であり、最小台数ではない。
イ：3台では1台故障時に2台となり、処理能力は500件/秒で不足する。
ウ：4台なら1台故障後も3台で750件/秒となり、600件/秒を上回る。
エ：5台でも要件は満たすが、最小台数ではない。
総合解説：正常時だけでなく障害時の必要容量を条件に入れて台数を決める。ここでは故障後3台必要なので、通常時は4台が最小となる。

対象：2026年度 / 基準日 2026-09-01

0012 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：基礎

重要なアーキテクチャ判断について、採用案だけでなく代替案・理由・前提条件も記録する主な利点はどれか。

ア．設計レビューを不要にできる。
イ．非機能要件を記載しなくてよくなる。
ウ．実装者がインタフェース仕様を自由に変更できる。
エ．将来の変更時に当時の判断根拠を確認し、前提変化を評価できる。

答え・解説 正答：エ

ア：記録はレビューの代替ではない。
イ：要件や制約の記録は引き続き必要である。
ウ：重要な契約を自由変更させる趣旨ではない。
エ：判断の背景を残すことで、前提が変わった際に再評価しやすくなる。
総合解説：方式決定にはトレードオフがあるため、根拠・前提・却下案を追跡可能にすると保守や再設計の質が上がる。

対象：2026年度 / 基準日 2026-09-01

0013 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：応用

負荷試験でCPU使用率は35%、DB待機時間は小さい一方、外部API応答待ちが全応答時間の80%を占めた。最初に検討すべき改善はどれか。

ア．外部API呼出しのタイムアウト、並列化、キャッシュ、非同期化などを要件と整合させて評価する。
イ．画面の色を変更する。
ウ．CPUを2倍に増強する。
エ．DBインデックスを全削除する。

答え・解説 正答：ア

ア：測定で支配的な待機要因が外部APIなので、その依存を中心に改善案を評価する。
イ：UI色は応答時間の主原因ではない。
ウ：CPUが主な待ち要因ではないため効果は限定的と考えられる。
エ：DB待機が小さいため、インデックス削除は根拠がない。
総合解説：性能設計では推測ではなく測定値からボトルネックを特定し、要求を満たす範囲で方式を改善する。

対象：2026年度 / 基準日 2026-09-01

0014 システムアーキテクチャ > 方式選定・アーキテクチャ・トレードオフ ｜ 難易度：標準

重要システムでRTOが30分、RPOが5分と定められた。バックアップを1日1回だけ別拠点へ搬送する方式の評価として適切なものはどれか。

ア．RPOが短いほどバックアップ頻度を下げるべきである。
イ．RPO 5分を満たすとは限らないため、より短い間隔の複製・バックアップを検討する。
ウ．RTOとRPOは画面応答時間の指標なので無関係である。
エ．日次バックアップなら必ずRTO 30分を満たす。

答え・解説 正答：イ

ア：短いRPOほど一般に更新データをより頻繁に保全する必要がある。
イ：日次では最大約1日分のデータ損失があり得るためRPO 5分には不足する。
ウ：RTOは復旧時間目標、RPOは許容データ損失時点の目標である。
エ：復元・切替時間を確認しないとRTO達成は判断できない。
総合解説：DR方式はRTO・RPOなどの事業要求から逆算し、複製頻度、待機系、切替手順、訓練まで設計する。

対象：2026年度 / 基準日 2026-09-01

0015 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：基礎

サブシステム分割の原則として最も適切なものはどれか。
ア．共通データを扱う全処理を一つの巨大サブシステムへ集める。
イ．変更頻度に関係なく全機能を相互参照させる。
ウ．関連の強い機能をまとめ、サブシステム間の依存を小さくする。
エ．全機能を均等な行数になるよう機械的に分ける。

答え・解説 正答：ウ

ア：巨大化すると責務が混在し変更影響が広がる。
イ：相互参照は結合度を高める。
ウ：高凝集・低結合を目指すことで変更影響を局所化しやすい。
エ：コード量の均等化は業務責務の境界を保証しない。
総合解説：サブシステムは責務や変更単位を意識し、内部の関連を強く、外部依存を必要最小限にする。

対象：2026年度 / 基準日 2026-09-01

0016 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：標準

価格計算ルールだけが毎月変更され、受注登録や在庫引当はほとんど変わらない。保守性を高める分割として適切なものはどれか。
ア．全機能を一つの巨大関数に統合する。
イ．価格計算を全画面へコピーする。
ウ．変更のたびにDBスキーマ全体を作り直す。
エ．価格計算ロジックを独立した責務として分離し、安定機能との契約を明確にする。

答え・解説 正答：エ

ア：巨大関数では影響範囲が広がる。
イ：重複実装は不整合と保守コストを増やす。
ウ：業務ルール変更だけで全DB再設計するのは不適切である。
エ：変化しやすい責務を局所化すれば変更影響を抑えやすい。
総合解説：変更軸を考慮した境界設計は、頻繁に変わる機能を安定部分から分離し、独立変更をしやすくする。

対象：2026年度 / 基準日 2026-09-01

**0017** システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：標準

工場内で毎秒大量のセンサデータを解析し、異常時は100ミリ秒以内に装置停止が必要である。クラウド接続は不安定なことがある。適切な機能配置はどれか。

ア．停止判断に必要な処理を工場側へ配置し、集計・長期分析はクラウドと分担する。
イ．センサ値を一日一回だけUSBで送る。
ウ．停止判断は人が翌日に行う。
エ．全解析と停止判断を遠隔クラウドだけで行う。

**答え・解説** 正答：ア

ア：厳しい応答要件をエッジ側で処理し、クラウドと役割分担するのが合理的である。
イ：即時性要件を満たさない。
ウ：100ミリ秒以内という要件に反する。
エ：通信遅延・断絶でリアルタイム停止要件を満たせない恐れがある。
総合解説：機能配置は応答時間、データ量、通信信頼性、運用性などを考慮して決める。

対象：2026年度 / 基準日 2026-09-01

**0018** システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：標準

複数サブシステムが互いの管理テーブルを直接更新している。最も大きな設計上の問題はどれか。

ア．DBを共有すればインタフェース設計は一切不要になる。
イ．サブシステムごとのデータ責務が曖昧になり、スキーマ変更の影響が波及しやすい。
ウ．SQLを利用すると必ず性能が低下する。
エ．テーブル数が増えるほど結合度は必ず下がる。

**答え・解説** 正答：イ

ア：共有DBでも契約・権限・責務の設計は必要である。
イ：データ所有境界を越えた直接更新は強い結合を生みやすい。
ウ：SQL自体が問題なのではなく依存の持ち方が問題である。
エ：テーブル数だけで結合度は決まらない。
総合解説：サブシステム境界を保つには、データ所有者を明確にし、必要な連携は定義されたインタフェース経由にするなどの方針を検討する。

対象：2026年度 / 基準日 2026-09-01

0019 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：基礎

サブシステム間インタフェース仕様として、最も不足すると相互接続性に直結する項目はどれか。
ア．開発者の座席番号
イ．プロジェクトの懇親会日程
ウ．データ項目・型・必須条件・エラー時の扱い
エ．担当者の好みのエディタ

答え・解説 正答：ウ

ア：座席番号はインタフェース仕様ではない。
イ：懇親会日程はシステム接続条件ではない。
ウ：データ契約と異常系は相互接続に直接必要である。
エ：開発ツールの好みは外部契約ではない。
総合解説：インタフェースでは入力・出力、意味、形式、タイミング、異常時処理、互換性などを明確にする。

対象：2026年度 / 基準日 2026-09-01

0020 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：標準

サブシステムAがBを直接呼び、BがCを直接呼び、Cが再びAを直接呼ぶ循環依存がある。保守性向上の方針として適切なものはどれか。
ア．循環を増やして均等にする。
イ．全サブシステムが全サブシステムを直接呼ぶ。
ウ．循環依存は変更影響に関係しないので放置する。
エ．依存方向を整理し、共通契約やイベントなどを使って循環を解消できないか検討する。

答え・解説 正答：エ

ア：循環を増やすと結合が強まる。
イ：全結合は変更影響をさらに拡大する。
ウ：循環依存はビルド・デプロイ・変更容易性へ影響する。
エ：循環依存は独立変更やテストを難しくするため依存方向の整理が有効である。
総合解説：境界間の依存方向を明確にし、循環を避けることで構成要素の独立性を高める。

対象：2026年度 / 基準日 2026-09-01

0021 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：応用

複数業務で「通知」機能を共通化する案がある。ただし、ある業務だけは厳格な順序保証と監査証跡が必要である。適切な設計判断はどれか。

ア．共通部分と業務固有要件を分離し、必要なら拡張点や別経路を設ける。
イ．監査要件を削除して共通化する。
ウ．通知を全業務ごとに完全コピーし、共通要素を一切持たない。
エ．全業務を最も厳しい仕様へ一律に合わせる以外に選択肢はない。

答え・解説 正答：ア

ア：共通責務と差異を明示して設計すると、再利用と固有要件を両立しやすい。
イ：要求を満たさない共通化は不適切である。
ウ：完全重複は保守負担を増やす。
エ：一律の過剰仕様はコスト増を招く可能性がある。
総合解説：共通化は同じ責務をまとめる一方、差異や将来変更を無視すると複雑化する。共通点と変動点を区別する。

対象：2026年度 / 基準日 2026-09-01

0022 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：標準

多数の利用システムがあるAPIで、新しい任意項目を追加する。互換性を保ちやすい変更はどれか。

ア．既存フィールドを削除して同名で別型にする。
イ．既存利用者が無視できる任意項目として追加し、契約上の互換性を確認する。
ウ．全レスポンス形式を予告なく変更する。
エ．既存必須項目の意味を逆転させる。

答え・解説 正答：イ

ア：削除・型変更は互換性を損なう。
イ：任意追加は一般に後方互換を維持しやすいが、利用側仕様の確認は必要である。
ウ：無告知の全面変更は利用者へ大きな影響を与える。
エ：意味変更は既存利用者を破壊する。
総合解説：外部契約は利用者数が増えるほど変更コストが高くなるため、互換性、版管理、廃止手順を設計する。

対象：2026年度 / 基準日 2026-09-01

0023 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：応用

「受注確定」と「在庫引当」を別サブシステムに分ける。受注確定時に在庫引当が失敗する可能性がある。方式設計で必要な検討はどれか。

ア．障害時は手順を定めず担当者の判断に任せる。
イ．分割したので失敗時の業務整合性は考えなくてよい。
ウ．跨る処理の整合性、補償、再試行、状態遷移を業務要件に応じて定める。
エ．両サブシステムのDBを利用者端末から直接更新する。

答え・解説 正答：ウ

ア：再現性ある復旧には設計された手順が必要である。
イ：境界を分けても業務整合性の責任は残る。
ウ：分散した処理は部分失敗を前提に業務状態の回復策を設計する必要がある。
エ：直接更新は境界と統制を崩す。
総合解説：サブシステム分割では技術境界だけでなく、業務トランザクションや失敗時の整合性を合わせて設計する。

対象：2026年度 / 基準日 2026-09-01

0024 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：標準

カメラ1台が毎秒20MBの生データを生成し、現地解析後は1秒当たり20KBの判定結果だけを送ればよい。回線帯域が限られる場合の配置として適切なものはどれか。

ア．帯域制約を無視して設計する。
イ．判定結果を画像に変換して容量を増やして送る。
ウ．生データを全て遠隔へ送り現地では何もしない。
エ．解析機能を現地側へ置き、必要な結果や選別データだけを送る。

答え・解説 正答：エ

ア：制約を無視すると運用時に要求を満たせない。
イ：不要な増量は制約に逆行する。
ウ：毎秒20MBを送るため帯域負荷が大きい。
エ：現地処理で転送量を大幅に減らせ、帯域制約に適合しやすい。
総合解説：データ量と処理場所の関係はアーキテクチャに大きく影響する。転送前に集約・解析できるかを検討する。

対象：2026年度 / 基準日 2026-09-01

0025 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：基礎

AとBの二つのサブシステムが、それぞれ独自の「顧客ランク判定」ロジックを保持している。最も懸念されることはどれか。

ア．ルール変更時に片方だけ更新され、判定結果が不一致になる。
イ．処理が二つあるので必ず可用性が向上する。
ウ．同じロジックを重複させるほど保守コストは下がる。
エ．データ型の定義が不要になる。

答え・解説 正答：ア

ア：同一業務ルールの重複実装は変更漏れによる不整合を招きやすい。
イ：単なる重複は冗長化としての可用性設計ではない。
ウ：重複箇所が増えるほど保守対象が増える。
エ：データ契約は引き続き必要である。
総合解説：業務ルールの所有責務を明確にし、重複実装を避けることで一貫性と保守性を高める。

対象：2026年度 / 基準日 2026-09-01

0026 システムアーキテクチャ > サブシステム分割・機能配置・インタフェース ｜ 難易度：標準

同期APIで外部サービスを呼ぶ際、タイムアウトを設定しない設計の問題として最も適切なものはどれか。

ア．ネットワーク遅延がゼロになる。
イ．外部応答が戻らない場合に呼出し元資源を長時間占有し、障害が連鎖する可能性がある。
ウ．タイムアウトがないほど必ずデータ整合性が高まる。
エ．外部サービス障害を自動的に修復できる。

答え・解説 正答：イ

ア：遅延自体をなくすものではない。
イ：待ち続ける呼出しが蓄積するとスレッドや接続等を枯渇させる恐れがある。
ウ：待ち時間の無制限化は整合性保証とは別問題である。
エ：タイムアウト不設定に修復効果はない。
総合解説：外部インタフェースは正常系だけでなく、タイムアウト、再試行、遮断、フォールバックなど異常系を設計する。

対象：2026年度 / 基準日 2026-09-01

0027 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：基礎

利用者がOSのパッチ適用やミドルウェア構成まで自ら管理したい場合、一般に最も選択自由度が高いクラウドサービスモデルはどれか。

ア．SaaS
イ．PaaS
ウ．IaaS
エ．業務委託だけで構成するBPO

答え・解説 正答：ウ

ア：SaaSはアプリケーションまでサービス提供側が管理する範囲が広い。
イ：PaaSは実行基盤の管理をサービス側へ委ねる範囲が大きい。
ウ：IaaSでは仮想サーバ等の基盤を利用し、OS以上を利用者が管理する構成を取りやすい。
エ：BPOはクラウドサービスモデルの分類ではない。
総合解説：サービスモデルごとに利用者と提供者の責任分界が異なるため、必要な制御範囲と運用負担を比較する。

対象：2026年度 / 基準日 2026-09-01

0028 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：標準

小規模チームが高可用DBを短期間で構築し、OS保守負担を減らしたい。適切な選択肢はどれか。

ア．バックアップを実施しないことで運用を簡素化する。
イ．可用性要件を削除して単一PCへ保存する。
ウ．DBサーバを全て自作ハードウェアで構築する。
エ．要件に合うマネージドDBを候補にし、機能制約・移行性・費用を評価する。

答え・解説 正答：エ

ア：運用負担軽減のために保全を捨てるのは要件違反となる。
イ：可用性要求を勝手に削除できない。
ウ：目的に対して設備・OS運用負担が大きい。
エ：マネージドサービスは運用作業を減らせる一方、制約やロックイン等を評価して採用する必要がある。
総合解説：クラウドではサービスに任せられる範囲と自組織が負う責任を明確にし、TCOや移行性も含めて比較する。

対象：2026年度 / 基準日 2026-09-01

0029 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：標準

機密データは社内設備から出せないが、公開Web部分はアクセス変動が大きい。適切な構成案はどれか。

ア．機密データ処理をオンプレミスに置き、公開Webをクラウドへ置くなど、境界と接続を設計したハイブリッド構成を検討する。
イ．機密データ制約を無視して全て外部公開する。
ウ．公開Webを社内ネットワークからのみアクセス可能にする。
エ．全てを必ず単一の社内サーバ1台へ集約する。

答え・解説 正答：ア

ア：制約の異なる領域を分け、連携・認証・監視を設計する方法が現実的である。
イ：機密保持要件に反する。
ウ：公開Webという業務要件に反する。
エ：変動負荷への対応や単一障害点の課題が残る。
総合解説：配置制約、伸縮性、運用、セキュリティを考慮し、必要ならオンプレミスとクラウドを組み合わせる。

対象：2026年度 / 基準日 2026-09-01

0030 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：基礎

1台の物理ホスト上に複数VMを配置する場合の注意点として適切なものはどれか。

ア．VMが分かれていれば物理ホスト障害の影響は必ず一つのVMだけに限られる。
イ．同一ホスト障害で複数VMが同時停止し得るため、可用性要件に応じて配置分散を検討する。
ウ．仮想化するとCPUやメモリの容量設計は不要になる。
エ．仮想化すればバックアップは不要になる。

答え・解説 正答：イ

ア：共有する物理ホストは共通障害点になり得る。
イ：論理分離と物理障害範囲は別なので、ホスト分散が必要な場合がある。
ウ：物理資源は有限で容量管理が必要である。
エ：データ保全や復旧設計は引き続き必要である。
総合解説：仮想化は資源利用効率や柔軟性を高めるが、共有基盤の障害影響や資源競合を考慮する。

対象：2026年度 / 基準日 2026-09-01

0031 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：基礎

一般的なコンテナ方式の特徴として最も適切なものはどれか。
ア．ネットワーク機能を利用できない。
イ．各コンテナが必ず独立した物理CPUを専有する。
ウ．ホストOSのカーネルを共有しつつ、プロセスやファイルシステム等を分離して実行する。
エ．OSを一切必要としない。

答え・解説 正答：ウ

ア：コンテナでもネットワークを利用できる。
イ：物理CPU専有はコンテナの一般的条件ではない。
ウ：コンテナはホストカーネルを共有する実装が一般的で、VMより軽量に起動しやすい。
エ：ホスト側OS/カーネル等の実行基盤は必要である。
総合解説：コンテナは配置・起動の柔軟性がある一方、隔離境界、永続化、イメージ管理、オーケストレーションなどを設計する必要がある。

対象：2026年度 / 基準日 2026-09-01

0032 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：標準

クラウド上で同一地域内の独立した複数障害区画へインスタンスを分散配置する主な狙いはどれか。
ア．全てのネットワーク遅延をゼロにする。
イ．アプリケーションの不具合を自動的に修正する。
ウ．データバックアップを不要にする。
エ．単一障害区画の停止によるサービス全停止リスクを下げる。

答え・解説 正答：エ

ア：分散しても通信遅延は存在する。
イ：ソフトウェア欠陥は別の対策が必要である。
ウ：可用性分散とデータ保全は別の論点である。
エ：障害ドメインを分けることで局所障害の影響を抑えやすい。
総合解説：クラウドの物理・論理障害ドメインを理解し、可用性目標に応じて配置を分散する。

対象：2026年度 / 基準日 2026-09-01

0033 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：標準

Webサーバを負荷に応じて増減したいが、各サーバのローカルメモリだけに利用者セッションを保持している。問題と改善策の組合せとして適切なものはどれか。

ア．サーバ増減や振分けでセッションを失う可能性があるため、共有ストア等へ外出しする方法を検討する。
イ．ローカル保持のままサーバを毎秒削除する。
ウ．セッションを画像ファイルへ変換すれば必ず解決する。
エ．負荷分散装置を削除すれば水平分散しやすくなる。

答え・解説 正答：ア

ア：状態がインスタンスに固定されると伸縮や障害復旧が難しくなるため、状態管理方式を見直すのが有効である。
イ：削除頻度を上げるほど状態喪失が増える。
ウ：形式変更では配置依存は解消しない。
エ：負荷分散は水平分散を支援する構成要素である。
総合解説：水平スケールを前提とする場合、サーバ固有状態を減らす、外部状態ストアを使うなどの設計を検討する。

対象：2026年度 / 基準日 2026-09-01

0034 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：基礎

クラウドサービスを採用した場合の運用責任について適切な説明はどれか。

ア．クラウドを使えば利用者側のセキュリティ設定責任は全て消える。
イ．サービスモデルと契約に応じて、提供者と利用者の責任範囲を明確にする必要がある。
ウ．提供者がバックアップ機能を持つなら復旧試験は不要である。
エ．クラウドでは監視を行えない。

答え・解説 正答：イ

ア：利用者側のID、権限、設定、データ管理などの責任は残る。
イ：責任共有を明確にしないと統制漏れが生じる。
ウ：機能の存在と復旧可能性の確認は別である。
エ：クラウドでも監視設計は可能かつ必要である。
総合解説：クラウド採用では「何を任せ、何を自組織が担うか」を明確にし、運用・セキュリティ・復旧手順へ落とし込む。

対象：2026年度 / 基準日 2026-09-01

0035 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：応用

特定クラウド固有の高機能サービスを使えば開発期間を半減できる一方、他環境への移行には大幅な改修が必要となる。適切な判断はどれか。

ア．固有サービスは常に禁止する。
イ．短期開発効果だけで採用し、将来移行は考えない。
ウ．事業上の便益、TCO、移行可能性、代替手段、契約条件を比較し、許容できる依存か判断する。
エ．ロックインはクラウドに存在しない。

答え・解説 正答：ウ

ア：依存自体が悪いのではなく便益との比較が必要である。
イ：将来コストや事業継続を無視した判断になる。
ウ：依存による便益と退出コストを明示してトレードオフを判断するのが適切である。
エ：固有APIやデータ形式などによる移行負担は存在し得る。
総合解説：クラウド固有機能は生産性を高めることがあるため、単純回避ではなく、移行・代替・契約終了時の影響を含めて判断する。

対象：2026年度 / 基準日 2026-09-01

0036 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：標準

新サービスの需要予測誤差が大きく、開始直後の利用者数が読めない。初期投資を抑えつつ増減へ対応したい場合の考え方として適切なものはどれか。

ア．容量監視を行わず自動拡張だけに依存する。
イ．需要が増えても構成を変更しない。
ウ．最大需要を仮定して最初から固定設備を大量購入する以外にない。
エ．伸縮可能なクラウド資源を候補とし、性能・費用・制約を監視しながら調整する。

答え・解説 正答：エ

ア：自動化にも閾値・費用上限・失敗時対応の監視が必要である。
イ：需要増に追随できない。
ウ：需要不確実性が高い場合は過剰投資リスクがある。
エ：伸縮性を活かしつつ監視と上限設定を行う方法が適合しやすい。
総合解説：需要不確実性、費用モデル、伸縮速度、サービス制約を比較して基盤方式を選ぶ。

対象：2026年度 / 基準日 2026-09-01

0037 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：標準

大量の画像推論をリアルタイム処理する必要があり、CPUだけでは性能要件を満たせない。適切な方式検討はどれか。

ア．GPU等のアクセラレータを候補にし、性能、費用、開発・運用制約をベンチマークで比較する。
イ．画像推論を廃止する。
ウ．性能要件を文書から削除する。
エ．性能測定を行わずCPU台数だけ無制限に増やす。

答え・解説 正答：ア

ア：ワークロードに適した計算資源を実測で比較するのが適切である。
イ：業務要求を勝手に削除できない。
ウ：要求削除は方式検討ではない。
エ：測定せず拡張するのは費用対効果を判断できない。
総合解説：ハードウェア構成は処理特性に合わせ、ベンチマークや将来容量、運用スキルを含めて評価する。

対象：2026年度 / 基準日 2026-09-01

0038 システムアーキテクチャ > ハードウェア・ソフトウェア・クラウド構成 ｜ 難易度：応用

外部クラウド管理サービスが長時間利用不能でも、重要設備の安全停止だけは現地で継続できることが要求された。構成として適切なものはどれか。

ア．安全停止ロジックも全て外部サービスへ依存させる。
イ．外部接続断を想定し、安全停止に必要な判断・設定・操作を現地側でも実行できるようにする。
ウ．外部停止時は設備を無制御で運転し続ける。
エ．要求を満たせないことを記録せず運用開始する。

答え・解説 正答：イ

ア：外部障害が安全機能へ直結する。
イ：重要機能に必要な自律性を確保する構成が要求に合う。
ウ：安全性を損なう。
エ：要求未達のまま稼働させるのは不適切である。
総合解説：重要情報・重要設備を扱うシステムでは、外部依存の途絶を想定し、自律的に最低限の機能を維持できるかを検討する。

対象：2026年度 / 基準日 2026-09-01

0039 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：基礎

システム要件をソフトウェア要件へ割り当てる際の基本的な考え方として適切なものはどれか。
ア．性能要件は実装後まで割当てない。
イ．要件と設計要素の対応関係は記録しない。
ウ．各要件をどの構成要素が実現・検証するかを明確にする。
エ．全要件を必ずアプリケーションソフトウェアだけに割り当てる。

答え・解説 正答：ウ

ア：性能は方式選定に影響するため早期配分が必要である。
イ：追跡関係がないと変更影響や検証が難しくなる。
ウ：責任主体と検証対象を明確にすることで要件漏れを防ぎやすい。
エ：要件によってはハードウェア、ネットワーク、運用などが担う。
総合解説：システム要件を構成要素へ配分し、設計・実装・テストまで追跡可能にする。

対象：2026年度 / 基準日 2026-09-01

0040 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：標準

「画面応答2秒以内」というシステム要件がある。適切な要件配分はどれか。
ア．2秒の根拠を消して設計者へ自由に任せる。
イ．性能測定は本番開始後に初めて行う。
ウ．画面だけに2秒要件を書き、DBや外部APIの時間は無視する。
エ．ネットワーク、アプリ、DB、外部連携などの時間予算へ分解し、全体で2秒を満たすよう設計する。

答え・解説 正答：エ

ア：要求を消すと合否を判断できない。
イ：早期の性能設計・検証が必要である。
ウ：全体応答は複数要素の合計なので一部だけでは管理できない。
エ：エンドツーエンド要件を構成要素の性能予算へ配分するのが有効である。
総合解説：システムレベルの非機能要求は構成要素に配分し、合成したときに全体要件を満たすようにする。

対象：2026年度 / 基準日 2026-09-01

0041 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：標準

機密データへの不正アクセスを防止するシステム要件の割当てとして適切なものはどれか。
ア．認証、認可、ネットワーク制御、暗号化、監査など必要な対策を複数層へ配分する。
イ．DB管理者にはアクセス制御を設定しない。
ウ．監査ログを残さない。
エ．アプリのログイン画面だけに全責任を持たせる。

答え・解説 正答：ア

ア：要求を構成要素へ適切に配分する多層防御が有効である。
イ：最小権限等の統制が必要である。
ウ：監査性の要求がある場合は記録が必要である。
エ：単一対策に依存すると迂回経路や内部脅威へ弱い。
総合解説：セキュリティ要件は一つのモジュールだけでなく、複数の技術・運用要素へ責任を配分することが多い。

対象：2026年度 / 基準日 2026-09-01

0042 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：標準

データ保存期間が1年から7年へ変更された。最初に確認すべき設計影響はどれか。
ア．保存量の増加は容量へ影響しないと仮定する。
イ．保存を担うストレージ、DB、バックアップ、検索性能、アーカイブ、運用手順への割当てを追跡する。
ウ．画面の配色だけを変更する。
エ．要件IDを削除して影響調査を省略する。

答え・解説 正答：イ

ア：容量・性能・費用へ影響する可能性が高い。
イ：保存要件に対応する構成要素を追跡すれば影響範囲を体系的に確認できる。
ウ：保存期間変更と直接関係しない。
エ：追跡性を失うと見落としが増える。
総合解説：要件と設計要素のトレーサビリティは、変更時の影響分析に重要である。

対象：2026年度 / 基準日 2026-09-01

0043 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：応用

月間可用性99.99%が必要なサービスで、アプリは自動再起動できるがDBが単一構成である。評価として適切なものはどれか。

ア．可用性はソフトウェア要件では扱えない。
イ．アプリが自動再起動するので全体可用性は自動的に99.99%になる。
ウ．エンドツーエンドで可用性を評価し、DBを含む各構成要素の冗長化・復旧時間を配分する。
エ．DB障害は可用性に含めない。

答え・解説 正答：ウ

ア：ソフトウェアにも再試行、フェイルオーバ等の責任を配分できる。
イ：全体は弱い構成要素の影響を受ける。
ウ：サービス経路全体で要求を分解・配分して設計する必要がある。
エ：DB停止もサービス停止要因である。
総合解説：全体可用性は構成要素の可用性と依存関係で決まるため、局所対策だけでなくシステム全体で評価する。

対象：2026年度 / 基準日 2026-09-01

0044 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：標準

顧客住所の郵便番号と都道府県の整合性を保証したい。適切な設計はどれか。

ア．帳票出力後に人が気付けばよいのでシステムでは検証しない。
イ．各画面が独自ルールを実装しルール共有をしない。
ウ．DBには不正値も無条件で保存する。
エ．入力時・取込時・更新時などデータが変更される境界で検証責務を定める。

答え・解説 正答：エ

ア：品質保証を人の偶然の発見に依存する。
イ：重複ルールは不一致を招く。
ウ：整合性要件に反する。
エ：データが生成・変更されるポイントへ検証責務を割り当てるのが有効である。
総合解説：データ品質要求は入力、API、バッチ、DB制約など適切な層へ割り当て、一貫したルールにする。

対象：2026年度 / 基準日 2026-09-01

0045 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：基礎

「障害原因を10分以内に特定できるようにする」という運用要件をソフトウェアへ反映する例として適切なものはどれか。

ア．相関ID付きログやメトリクスを出力し、監視から追跡できるようにする。
イ．例外を全て握りつぶして記録しない。
ウ．ログ形式をモジュールごとに毎回変える。
エ．障害時は本番環境のソースを直接書き換える。

答え・解説 正答：ア

ア：観測可能性を設計へ組み込むことで原因追跡を支援できる。
イ：情報が残らず原因分析が困難になる。
ウ：形式不統一は分析を難しくする。
エ：統制された診断・変更手順が必要である。
総合解説：運用性は運用部門だけの課題ではなく、ログ、監視、診断、設定変更などソフトウェア機能へ配分される。

対象：2026年度 / 基準日 2026-09-01

0046 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：標準

重要操作について「誰が・いつ・何をしたか」を後から確認できることが要求された。適切な割当てはどれか。

ア．全操作を匿名化して追跡不能にする。
イ．アプリケーションに識別情報付き監査ログを出力させ、保管・保護・検索の責務も定める。
ウ．利用者名だけを画面に表示しログは残さない。
エ．監査ログを一般利用者が自由に削除できるようにする。

答え・解説 正答：イ

ア：責任追跡性を失う。
イ：要求を満たすには生成だけでなく保全・参照まで設計する必要がある。
ウ：後から操作事実を追跡できない。
エ：改ざん防止・アクセス制御が必要である。
総合解説：監査要求はアプリ、認証基盤、ログ保管、運用権限など複数要素へ配分する。

対象：2026年度 / 基準日 2026-09-01

0047 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：基礎

要件トレーサビリティ表で「対応設計要素なし」となっている要件を確認する主な理由はどれか。
ア．テストケースを減らすため。
イ．設計者の作業時間だけを評価するため。
ウ．実装漏れや設計漏れの可能性を検出するため。
エ．要件を自動的に削除するため。

答え・解説 正答：ウ

ア：要件に対応する検証はむしろ必要になる。
イ：目的は要求実現の完全性確認である。
ウ：未割当て要件は実現責任が不明であり、漏れの兆候となる。
エ：削除には合意された変更手続が必要である。
総合解説：要件から設計・実装・試験への対応関係を確認し、未割当てや過剰設計を検出する。

対象：2026年度 / 基準日 2026-09-01

0048 ソフトウェア要件・方式設計 > ソフトウェア要件への割当て ｜ 難易度：応用

「全検索結果を常に最新にする」と「外部回線断でも30分は検索を継続する」という要件が競合する可能性がある。適切な対応はどれか。
ア．片方を設計者が黙って削除する。
イ．回線断時も外部DBへ必ず接続できると仮定する。
ウ．両方を無条件に満たせると仮定して設計を進める。
エ．許容データ鮮度、切断時の動作、優先順位を利害関係者と明確化し、キャッシュ等の責務へ配分する。

答え・解説 正答：エ

ア：要件変更には合意が必要である。
イ：障害時要件と矛盾する仮定である。
ウ：競合を放置すると実装段階で破綻する。
エ：要求の優先度と許容範囲を明確にして設計へ落とすのが適切である。
総合解説：要件配分では要求間の矛盾・競合を検出し、トレードオフを合意してから構成要素へ責任を割り当てる。

対象：2026年度 / 基準日 2026-09-01

0049 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：基礎

プレゼンテーション層、業務ロジック層、データアクセス層に分ける主な目的はどれか。
ア．関心事を分離し、変更影響を限定しやすくする。
イ．全層がDBへ自由に直接アクセスするため。
ウ．同じロジックを各層へ複製するため。
エ．テストを不可能にするため。

答え・解説 正答：ア

ア：役割を分離するとUI変更と業務ロジック変更などを独立させやすい。
イ：層を飛び越えた依存は分離の利点を損なう。
ウ：重複は保守性を悪化させる。
エ：分離はむしろ単体テストをしやすくする場合がある。
総合解説：レイヤ構造では各層の責務と依存方向を定め、関心事の分離を図る。

対象：2026年度 / 基準日 2026-09-01

0050 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：標準

業務ロジックが特定DB製品のAPIを直接呼んでおり、DB変更が業務コードへ広く波及している。改善方針として適切なものはどれか。
ア．テスト環境を廃止する。
イ．業務側が抽象化されたデータアクセス契約に依存し、製品固有実装を外側へ隔離する。
ウ．全業務コードへDB製品名をハードコードする。
エ．DB変更のたびに業務仕様も変更する。

答え・解説 正答：イ

ア：テスト廃止は改善にならない。
イ：技術詳細を抽象化の背後へ置けば業務ロジックへの変更影響を減らしやすい。
ウ：製品依存がさらに広がる。
エ：業務要求と技術製品は分離すべきである。
総合解説：安定した業務責務と変化しやすい技術詳細の依存関係を整理し、置換可能性を高める。

対象：2026年度 / 基準日 2026-09-01

0051 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：標準

複数機能が「注文確定」という事実を契機に、それぞれ通知・分析・ポイント付与を独立実行したい。追加機能も増える予定である。適切な方式はどれか。

ア．全処理を一つの画面イベントハンドラへ記述する。
イ．注文機能が全利用機能を個別に同期呼出しする以外にない。
ウ．注文確定イベントを発行し、必要な処理が購読するイベント駆動方式を検討する。
エ．注文DBを全機能が直接更新する。

答え・解説 正答：ウ

ア：巨大化して変更影響が広がる。
イ：利用機能が増えるほど注文側の結合が強まる。
ウ：イベント発行者が購読者を直接知らずに拡張しやすい。
エ：共有DB直接更新は責務境界を崩しやすい。
総合解説：イベント駆動は疎結合化に有効だが、順序、重複、失敗、追跡性を設計する必要がある。

対象：2026年度 / 基準日 2026-09-01

0052 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：基礎

MVCで、利用者入力を受け取り適切な処理へ橋渡しする役割に最も近いものはどれか。

ア．Database
イ．Model
ウ．View
エ．Controller

答え・解説 正答：エ

ア：DatabaseはMVCの三要素そのものではない。
イ：Modelは主に状態や業務データ・ロジックを扱う。
ウ：Viewは表示を担う。
エ：Controllerは入力を受け、モデルやビューとの調停を行う役割を持つ。
総合解説：MVCは表示、入力制御、モデルを分離する代表的な設計パターンである。

対象：2026年度 / 基準日 2026-09-01

0053 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：応用

決済API呼出しがタイムアウトし、実際には決済が成功していた可能性がある。呼出し元が無条件に同じ要求を再送すると二重決済の恐れがある。適切な設計はどれか。

ア．一意な処理キー等で同一要求を識別し、再送しても重複処理にならない仕組みを設ける。
イ．タイムアウトを全て無限待ちにする。
ウ．再送回数を無限にする。
エ．決済結果を記録しない。

答え・解説 正答：ア

ア：冪等キー等により重複要求を同一処理として扱える設計が有効である。
イ：無限待ちは資源枯渇や障害連鎖を招く。
ウ：無限再試行は外部負荷や重複リスクを高める。
エ：追跡不能になり復旧が困難になる。
総合解説：分散処理では通信結果が不確実になるため、再試行と冪等性を組み合わせて二重処理を防ぐ。

対象：2026年度 / 基準日 2026-09-01

0054 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：標準

接続先URLやタイムアウト値が環境ごとに異なる。適切な設計はどれか。

ア．ソースコード内に本番値を直接埋め込み、環境ごとにコードを改変する。
イ．設定を外部化し、環境ごとに安全に注入・管理できるようにする。
ウ．値を開発者の記憶だけで管理する。
エ．全環境で同じ接続先を使う。

答え・解説 正答：イ

ア：環境差分のためにコード変更が必要となり誤配布リスクが高まる。
イ：コードと環境設定を分離すると同一成果物を環境間で利用しやすい。
ウ：再現性・監査性がない。
エ：環境分離要件を満たせない。
総合解説：構成情報を適切に外部化し、秘密情報はアクセス制御された仕組みで扱う。

対象：2026年度 / 基準日 2026-09-01

0055 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：標準

DBの一時障害が発生した際、データアクセス層の内部例外をそのまま利用者画面へスタックトレース付きで表示している。改善として適切なものはどれか。

ア．例外を全て無視して成功扱いにする。
イ．障害ログを削除する。
ウ．内部例外を適切に変換・記録し、利用者には必要なエラー情報だけを返す。
エ．スタックトレースを常に全利用者へ表示する。

答え・解説 正答：ウ

ア：データ不整合や誤認を招く。
イ：原因追跡が困難になる。
ウ：責務境界でエラーを変換し、内部情報漏えいを避けつつ診断情報は記録する。
エ：内部構造や機密情報を漏らす恐れがある。
総合解説：例外処理は技術詳細の隠蔽、利用者通知、ログ、再試行可否を層の責務に応じて設計する。

対象：2026年度 / 基準日 2026-09-01

0056 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：標準

数十分かかる帳票生成をWeb要求から起動する。利用者が進捗確認でき、ブラウザ切断でも処理を継続させたい。適切な方式はどれか。

ア．ブラウザが閉じたら必ず処理も取り消す。
イ．処理状態をどこにも保存しない。
ウ．HTTP接続を数十分保持し続けるだけにする。
エ．ジョブを非同期キューへ登録し、ジョブIDで状態と結果を照会できるようにする。

答え・解説 正答：エ

ア：要件では切断後も継続が必要である。
イ：進捗や再開を確認できない。
ウ：長時間接続はタイムアウトや資源占有の問題がある。
エ：ジョブと対話処理を分離し、状態を永続管理すると要件を満たしやすい。
総合解説：長時間処理は非同期化し、受付・実行・状態・結果取得を分ける設計が有効である。

対象：2026年度 / 基準日 2026-09-01

0057 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：標準

複数地域で利用するシステムで、イベント発生時刻の比較・並べ替えが必要である。適切な設計はどれか。
ア．保存時刻のタイムゾーンを明確にし、基準時刻と表示ローカル時刻を区別する。
イ．各サーバが任意のローカル表記だけで保存する。
ウ．夏時間の存在を無視する。
エ．時刻を文字列の見た目だけで比較する。

答え・解説 正答：ア

ア：基準とタイムゾーンを明確にすると地域間比較や再現性が高まる。
イ：意味の異なるローカル時刻が混在する恐れがある。
ウ：地域によっては時刻ずれを生じる。
エ：形式によっては正しい時系列比較にならない。
総合解説：時刻は業務上の意味、タイムゾーン、精度、同期方式を設計し、保存と表示を区別する。

対象：2026年度 / 基準日 2026-09-01

0058 ソフトウェア要件・方式設計 > コンポーネント・ソフトウェア方式設計 ｜ 難易度：基礎

機能フラグを利用する目的として適切なものはどれか。
ア．フラグを永久に増やし続けても複雑性は増えない。
イ．コードを配布したまま機能の有効・無効を制御し、段階的公開などを行いやすくする。
ウ．テストを不要にする。
エ．アクセス制御を全て置き換える。

答え・解説 正答：イ

ア：古いフラグの残存は分岐複雑性を増やす。
イ：配布と公開を分離できるため段階リリース等に利用できる。
ウ：有効・無効双方の検証が必要である。
エ：認可とは別の目的である。
総合解説：機能フラグはリリース柔軟性を高めるが、所有者、期限、監視、削除を管理する必要がある。

対象：2026年度 / 基準日 2026-09-01

0059 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：基礎

業務パッケージ選定でFit & Gap分析を行う主な目的はどれか。
ア．全ての不足を必ず個別開発する。
イ．利用者数を調べずに価格だけで決める。
ウ．業務要件と標準機能の適合部分・不足部分を明確にする。
エ．パッケージのロゴデザインだけを比較する。

答え・解説 正答：ウ

ア：業務変更、代替運用、追加開発など複数の対応がある。
イ：費用以外の要件も評価する必要がある。
ウ：標準機能で満たせる範囲とギャップを把握し対応方針を決めるために行う。
エ：見た目だけでは業務適合性を判断できない。
総合解説：パッケージは標準機能への業務適合を確認し、ギャップへの対応コストと保守影響を評価する。

対象：2026年度 / 基準日 2026-09-01

0060 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：標準

ERP導入で現行業務をそのまま再現するため大量の独自カスタマイズを予定している。懸念として最も適切なものはどれか。
ア．独自変更が多いほど標準機能への追随は必ず容易になる。
イ．カスタマイズは保守費用に影響しない。
ウ．業務プロセスの見直しは絶対に行ってはならない。
エ．将来のバージョンアップ時に改修・検証負担が増える可能性がある。

答え・解説 正答：エ

ア：標準から離れるほど追随は難しくなる場合がある。
イ：保守・テスト・移行コストへ影響する。
ウ：パッケージ標準へ業務を合わせる選択も比較対象である。
エ：独自差分は更新時の競合や再検証対象を増やしやすい。
総合解説：パッケージ選定では導入時だけでなく、アップグレードや保守を含むTCOでカスタマイズを評価する。

対象：2026年度 / 基準日 2026-09-01

0061 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：基礎

OSSを製品へ組み込む際の基本的な確認として適切なものはどれか。
ア．利用形態と各OSSのライセンス条件を確認する。
イ．公開リポジトリにあれば無条件に著作権が放棄されているとみなす。
ウ．OSSにはライセンスが存在しない。
エ．利用開始後もライセンス条件を確認する必要はない。

答え・解説 正答：ア

ア：OSSごとに許諾条件があり、配布・改変等の利用形態との適合確認が必要である。
イ：公開されていることと権利放棄は同義ではない。
ウ：OSSもライセンスに基づき利用される。
エ：更新や依存関係変更時にも確認が必要になる。
総合解説：OSSは「無料だから自由」ではなく、ライセンス、セキュリティ、保守状況を組織として管理する。

対象：2026年度 / 基準日 2026-09-01

0062 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：標準

業務で長期利用するOSS候補を評価する観点として最も適切なものはどれか。
ア．最終更新が10年前でも依存関係を確認しない。
イ．最新コミット日、リリース頻度、脆弱性対応、コミュニティ、ライセンスなどを総合評価する。
ウ．星の数だけで決める。
エ．無償なら保守状況は確認しない。

答え・解説 正答：イ

ア：依存先の状態も供給網リスクになる。
イ：活動性・品質・セキュリティ・法務等を複数観点で見る必要がある。
ウ：人気指標だけでは企業利用の適合性は判断できない。
エ：保守停止は脆弱性や互換性リスクになる。
総合解説：OSS選定では機能適合だけでなく、保守継続性とサプライチェーンを評価する。

対象：2026年度 / 基準日 2026-09-01

0063 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：標準

成熟したWebフレームワークを採用する利点と注意点の組合せとして適切なものはどれか。
ア．採用すると脆弱性対応は不要になる。
イ．フレームワークは必ず自社開発より高価である。
ウ．定型機能を再利用できる一方、規約・更新・依存関係への追随が必要になる。
エ．採用すると設計判断が一切不要になる。

答え・解説 正答：ウ

ア：依存部品の脆弱性管理は必要である。
イ：費用は条件によって異なる。
ウ：生産性向上の反面、フレームワーク固有制約や更新管理が発生する。
エ：業務設計や非機能設計は残る。
総合解説：フレームワークは再利用性と標準化を提供するが、学習コスト・制約・ライフサイクルを評価する。

対象：2026年度 / 基準日 2026-09-01

0064 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：応用

競争優位に直結しない給与計算機能について、成熟したパッケージが法改正対応も含め提供されている。内製案との比較で適切な判断はどれか。
ア．初期価格だけで最安を選ぶ。
イ．パッケージならセキュリティ評価は不要とする。
ウ．必ず内製する。
エ．機能適合、法改正対応、導入・保守TCO、データ連携、ロックイン等を比較して決める。

答え・解説 正答：エ

ア：保守・移行費用などを見落とす。
イ：外部製品でもセキュリティ確認は必要である。
ウ：内製が常に最適とは限らない。
エ：業務適合とライフサイクルコストを総合評価するのが適切である。
総合解説：Make or Buyは差別化、機能適合、納期、スキル、TCO、継続性を比較するアーキテクチャ判断である。

対象：2026年度 / 基準日 2026-09-01

0065 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：標準

新規システムの予定利用期間は7年だが、候補ミドルウェアのベンダサポート終了が2年後である。適切な対応はどれか。

ア．移行・更新計画と費用を含めて評価し、別製品も比較する。
イ．サポート終了後は脆弱性が発生しないと仮定する。
ウ．保守部門には知らせない。
エ．サポート期限を無視して採用する。

答え・解説 正答：ア

ア：ライフサイクルを通じた保守可能性を評価する必要がある。
イ：サポート終了は脆弱性修正や互換性対応のリスクを高める。
ウ：運用計画との共有が必要である。
エ：運用期間中に保守不能となるリスクがある。
総合解説：製品・OSS・フレームワークはEOL/EOSを確認し、利用期間と更新ロードマップを整合させる。

対象：2026年度 / 基準日 2026-09-01

0066 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：基礎

多数のOSSライブラリを利用するシステムで、使用部品とバージョンを継続的に把握する主な理由はどれか。

ア．テストを不要にするため。
イ．脆弱性公表時に影響を受ける構成要素を迅速に特定するため。
ウ．ソースコードの行数を増やすため。
エ．全OSSを自動的に無償化するため。

答え・解説 正答：イ

ア：依存管理はテストの代替ではない。
イ：依存部品の可視化は脆弱性やライセンス影響の特定に役立つ。
ウ：行数増加は目的ではない。
エ：利用条件はライセンスごとに決まる。
総合解説：ソフトウェアサプライチェーンでは、使用部品・版・依存関係を把握し、脆弱性対応へつなげることが重要である。

対象：2026年度 / 基準日 2026-09-01

0067 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：標準

クラウドVM上にOSSのWebサーバを自組織で導入した。脆弱性修正について適切な説明はどれか。
ア．公開ポートが少なければ更新不要である。
イ．クラウド上なのでOSSの更新責任は自動的に提供者へ移る。
ウ．責任分界に従い、自組織が管理するOS・ミドルウェアの更新責任を確認する。
エ．OSSなら脆弱性は存在しない。

答え・解説 正答：ウ

ア：露出を減らしても既知脆弱性対応は必要である。
イ：IaaS等ではゲストOS以上を利用者が管理することが多い。
ウ：責任共有モデルに基づく確認が必要である。
エ：OSSにも脆弱性は発見される。
総合解説：クラウド利用とOSS利用の責任を重ね合わせ、誰が何を更新するかを運用設計で明確にする。

対象：2026年度 / 基準日 2026-09-01

0068 ソフトウェア要件・方式設計 > パッケージ・OSS・フレームワーク選定 ｜ 難易度：応用

SaaS型業務パッケージを選定する際、将来のサービス終了や乗換えに備えた確認として最も適切なものはどれか。
ア．導入時には退出を考えない。
イ．画面から見えるデータだけ存在すると仮定する。
ウ．契約終了時のデータ処理は提供者に無条件で任せ、条件を確認しない。
エ．データのエクスポート形式、取得可能範囲、API、契約終了時の保持・削除条件を確認する。

答え・解説 正答：エ

ア：退出設計を後回しにすると移行不能や高コストになる。
イ：内部履歴や添付等の取得範囲も確認が必要である。
ウ：契約条件と責任を明確にする必要がある。
エ：データ可搬性と終了条件はロックイン・事業継続へ直結する。
総合解説：製品選定では導入だけでなく、更新・終了・データ移行まで含むライフサイクルを評価する。

対象：2026年度 / 基準日 2026-09-01

0069 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：基礎

業務上の「顧客」「注文」「商品」とその関係を、特定DBMSの型や索引を決めずに整理する段階に最も近いものはどれか。

ア．概念データモデル
イ．物理データ配置
ウ．索引チューニング
エ．バックアップ媒体設計

答え・解説 正答：ア

ア：業務実体と関係を技術詳細から独立して整理する。
イ：物理配置は実装段階の事項である。
ウ：索引は物理設計で検討する。
エ：バックアップ媒体は運用・物理構成の事項である。
総合解説：データ設計は業務意味を表すモデルから論理構造、物理実装へ段階的に具体化する。

対象：2026年度 / 基準日 2026-09-01

0070 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：基礎

関係データベースで正規化を行う主な目的として最も適切なものはどれか。

ア．主キーをなくす。
イ．データの重複や更新時の不整合を減らす。
ウ．検索SQLを必ず一行にする。
エ．全テーブルを一つに統合する。

答え・解説 正答：イ

ア：識別に必要なキー設計は重要である。
イ：関数従属性等を整理して更新異常を減らすことが目的である。
ウ：SQLの記述量が目的ではない。
エ：一表化は重複を増やす場合がある。
総合解説：正規化はデータの意味と従属性を整理し、挿入・更新・削除時の異常を抑える。

対象：2026年度 / 基準日 2026-09-01

0071 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：標準

正規化された集計DBで、複雑な結合により月次レポートが締切に間に合わない。適切な対応はどれか。
ア．全制約を削除する。
イ．DBを使わず画面に全データを保存する。
ウ．測定結果と整合性維持方法を確認したうえで、集計表やマテリアライズ等の非正規化を候補にする。
エ．正規化は絶対なので性能要件を削除する。

答え・解説 正答：ウ

ア：制約削除はデータ品質を損なう。
イ：永続データ管理として不適切である。
ウ：性能と整合性のトレードオフを管理できるなら非正規化は選択肢となる。
エ：業務要求を満たすため方式を見直すべきである。
総合解説：論理設計の原則を踏まえつつ、物理設計では性能要件に応じて索引、パーティション、集計構造などを検討する。

対象：2026年度 / 基準日 2026-09-01

0072 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：標準

外部機関が発行する顧客番号が将来変更される可能性があり、履歴も保持したい。内部主キーの設計として適切な考え方はどれか。
ア．主キーを設けない。
イ．氏名を主キーにする。
ウ．変更可能な顧客番号だけを永続的な主キーに固定する。
エ．変更されない内部代理キーを識別子とし、顧客番号は業務属性として一意制約等を検討する。

答え・解説 正答：エ

ア：行の一意識別が難しくなる。
イ：同姓同名や氏名変更がある。
ウ：外部番号変更時に参照関係へ大きな影響が出る可能性がある。
エ：内部同一性と外部業務番号を分離すると変更に強くなる。
総合解説：キー設計では安定性、一意性、外部連携、履歴要件を考慮し自然キーと代理キーを使い分ける。

対象：2026年度 / 基準日 2026-09-01

0073 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：標準

商品価格の「現在値」だけでなく、過去の注文時点で適用された価格を再現する必要がある。適切なデータ設計はどれか。

ア．有効開始・終了等を持つ価格履歴を管理し、注文時点の価格を特定できるようにする。
イ．過去価格は担当者の記憶に頼る。
ウ．毎回同じ価格だったと仮定する。
エ．商品テーブルの価格を上書きし履歴を残さない。

答え・解説 正答：ア

ア：時点に対応する履歴を保持すれば監査・再現要件を満たせる。
イ：再現性と証拠性がない。
ウ：価格変更要件と矛盾する。
エ：上書きでは過去状態を再現できない。
総合解説：履歴設計では「いつの値を再現するか」を明確にし、有効期間や版、イベント履歴など適切な方式を選ぶ。

対象：2026年度 / 基準日 2026-09-01

0074 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：基礎

一人の顧客が複数の電話番号を持ち、番号ごとに種別と優先順位を管理する。適切な論理設計はどれか。

ア．顧客テーブルに電話1、電話2、電話3を固定列で無制限に追加する。
イ．顧客と電話番号を1対多の別エンティティとして表現する。
ウ．電話番号を一つの巨大文字列に連結する。
エ．電話番号を顧客名の一部として保存する。

答え・解説 正答：イ

ア：件数上限が固定され拡張性と正規化に問題がある。
イ：繰返し属性を別エンティティにすると件数と属性を柔軟に扱える。
ウ：検索・制約・更新が困難になる。
エ：意味の異なる属性を混在させる。
総合解説：繰返しや多値属性は、関係モデル上の別表へ分離し、主従関係や制約を明確にする。

対象：2026年度 / 基準日 2026-09-01

0075 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：標準

複数組織で住所データを連携する際、同じ「都道府県」を各組織が独自コードで表している。改善として適切なものはどれか。

ア．数値が同じなら同じ意味と仮定する。
イ．組織ごとに列名も値形式も毎回変更する。
ウ．共通語彙・コード体系または明示的な対応表を定め、意味と変換規則を共有する。
エ．コードの意味を文書化しない。

答え・解説 正答：ウ

ア：偶然同じ値でも意味が異なる可能性がある。
イ：変換負担と誤解が増える。
ウ：意味と表現を共有することで相互運用性を高められる。
エ：意味不明となり連携品質が落ちる。
総合解説：データ連携では項目名だけでなく、語彙、コード、型、制約、意味を共通化または対応付けする。

対象：2026年度 / 基準日 2026-09-01

0076 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：標準

数十億件のログを日付範囲で検索・削除する処理が多い。物理設計として有効な候補はどれか。

ア．全ログを一つの巨大テキスト列に連結する。
イ．索引を全て禁止する。
ウ．日付列を削除する。
エ．日付をキーにしたパーティショニングを検討する。

答え・解説 正答：エ

ア：構造化検索が困難になる。
イ：必要な索引まで禁止すると性能を損なう。
ウ：検索条件を失う。
エ：アクセスと保守単位が日付なら範囲分割が検索・削除運用に合いやすい。
総合解説：物理設計はデータ量、アクセスパターン、保持・削除単位に合わせて索引やパーティションを決める。

対象：2026年度 / 基準日 2026-09-01

0077 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：標準

更新頻度が非常に高いテーブルに多数の索引を追加する場合の注意点はどれか。
ア．検索性能は改善し得るが、更新時の索引メンテナンス負荷と容量が増える。
イ．索引は追加するほど更新性能も必ず向上する。
ウ．索引はディスク容量を使わない。
エ．索引があれば統計情報や実行計画は不要である。

答え・解説 正答：ア

ア：索引は読取りと書込みでトレードオフがある。
イ：更新時には索引更新が必要になる。
ウ：索引も記憶領域を消費する。
エ：最適化には統計や実行計画の確認が有用である。
総合解説：索引は実際の検索条件・選択度・更新負荷を測定して設計する。

対象：2026年度 / 基準日 2026-09-01

0078 データ・連携設計 > 論理・物理データ設計 ｜ 難易度：標準

配送サービスの注文処理に生年月日が不要であるにもかかわらず、将来使うかもしれないという理由だけで必須収集する案がある。設計上の考え方として適切なものはどれか。
ア．不要データほどアクセス権を広げる。
イ．目的に必要なデータ項目かを確認し、不要なら収集・保持しない方針を検討する。
ウ．取得できる項目は全て永久保存する。
エ．利用目的を決めずに収集する。

答え・解説 正答：イ

ア：不要データの権限拡大は危険である。
イ：不要データを持たないことはリスクと管理コストを減らす。
ウ：漏えい時の影響や管理負担が増える。
エ：目的不明の収集は設計・統制上不適切である。
総合解説：データモデルは業務目的、保持期間、アクセス主体を踏まえ、必要なデータだけを扱う。

対象：2026年度 / 基準日 2026-09-01

0079 データ・連携設計 > DB整合性・トランザクション設計 ｜ 難易度：基礎

トランザクションの原子性（Atomicity）を表す説明として適切なものはどれか。
ア．同時実行時にも直列実行と整合する性質を持つ。
イ．業務ルールに矛盾しない状態を保つ。
ウ．一連の更新が全て成功するか、失敗時は全て取り消される。
エ．コミット後の結果が障害後も保持される。

答え・解説 正答：ウ

ア：これは独立性に関する説明である。
イ：これは一貫性に関する説明である。
ウ：処理の全か無かを保証する性質が原子性である。
エ：これは耐久性に近い。
総合解説：ACIDは原子性、一貫性、独立性、耐久性からなり、トランザクション設計の基礎となる。

対象：2026年度 / 基準日 2026-09-01

0080 データ・連携設計 > DB整合性・トランザクション設計 ｜ 難易度：基礎

トランザクションT1が、T2の未コミット更新値を読み、その後T2がロールバックした。この現象は何か。
ア．ロストアップデート
イ．デッドロック
ウ．チェックポイント
エ．ダーティリード

答え・解説 正答：エ

ア：ロストアップデートは競合更新で一方の更新が失われる現象である。
イ：デッドロックは相互待ちで進行不能になる状態である。
ウ：チェックポイントは障害回復に用いる仕組みである。
エ：未コミット値を読む現象がダーティリードである。
総合解説：隔離レベルにより許容される同時実行現象が異なるため、業務整合性と性能を考慮して選ぶ。

対象：2026年度 / 基準日 2026-09-01

0081 データ・連携設計 > DB整合性・トランザクション設計 ｜ 難易度：標準

同じデータを同時更新する頻度は低いが、利用者が長時間編集するWeb画面で、DBロックを長時間保持したくない。適切な方式はどれか。

ア．版番号等を用い更新時に競合を検出する楽観ロックを検討する。
イ．画面を開いた瞬間から行ロックを数時間保持する。
ウ．競合が起きても後勝ちで無条件上書きする。
エ．更新前の値を確認しない。

答え・解説 正答：ア

ア：競合が少ない場合、更新時に版を照合して競合検出する方式が適する。
イ：長時間ロックは他利用者を待たせる。
ウ：更新消失が起きる。
エ：競合検知できない。
総合解説：排他方式は競合頻度、保持時間、利用者体験、整合性要求から選ぶ。

対象：2026年度 / 基準日 2026-09-01

0082 データ・連携設計 > DB整合性・トランザクション設計 ｜ 難易度：標準

T1は表A→表B、T2は表B→表Aの順にロックし、デッドロックが頻発している。改善として有効なものはどれか。

ア．ロックを取得したまま利用者入力を待つ。
イ．ロック取得順序を統一する。
ウ．全トランザクションで異なる順序をランダムにする。
エ．タイムアウトを無限にする。

答え・解説 正答：イ

ア：保持時間が長くなり競合を悪化させる。
イ：資源取得順序を統一すると循環待ちの条件を減らせる。
ウ：順序のばらつきはデッドロックを増やし得る。
エ：無限待ちは解消を遅らせる。
総合解説：デッドロックはロック順序、トランザクション短縮、検出・再試行などで対処する。

対象：2026年度 / 基準日 2026-09-01

**0083** データ・連携設計 > DB整合性・トランザクション設計 ｜ 難易度：応用

注文、在庫、配送が独立サービスで動作し、各サービスが別DBを持つ。長時間の業務処理を単一DBトランザクションで囲めない。適切な考え方はどれか。

ア．失敗時の状態は記録しない。
イ．一部失敗しても必ず成功扱いにする。
ウ．部分失敗を前提に状態遷移と補償処理を設計する。
エ．全サービスDBを一つの共有テーブルへ統合する以外にない。

**答え・解説** 正答：ウ

ア：復旧・再実行が困難になる。
イ：業務不整合を隠すことになる。
ウ：分散業務では各局所処理と補償を組み合わせる方式が選択肢となる。
エ：独立性やスケーラビリティを失う可能性がある。
総合解説：分散処理では原子的コミットが難しい場合、Saga等の補償型の設計を業務要件に応じて検討する。

対象：2026年度 / 基準日 2026-09-01

**0084** データ・連携設計 > DB整合性・トランザクション設計 ｜ 難易度：標準

同じ外部注文IDを二重登録してはいけない。アプリ側の事前検索だけでは競合時に重複する可能性がある。追加対策として適切なものはどれか。

ア．重複時は運用で偶然気付くのを待つ。
イ．外部注文ID列を削除する。
ウ．全注文を同じIDにする。
エ．DBに外部注文IDの一意制約を設定する。

**答え・解説** 正答：エ

ア：競合条件を防止できない。
イ：重複判定の根拠を失う。
ウ：一意性要件に反する。
エ：DB制約で最終的な一意性を強制できる。
総合解説：重要な整合性ルールはアプリだけに依存せず、DB制約等で可能な範囲を強制する。

対象：2026年度 / 基準日 2026-09-01

0085 データ・連携設計 > DB整合性・トランザクション設計 ｜ 難易度：応用

DBへの注文登録後にメッセージを発行する処理で、DBコミット直後にプロセスが停止するとメッセージだけ発行されないことがある。対策として適切なものはどれか。

ア．DB更新と同一トランザクションで送信予定イベントをOutbox表へ記録し、別処理が確実に配信する方式を検討する。
イ．メッセージ送信失敗を無視する。
ウ．DBコミットをなくす。
エ．毎回全注文を再送し重複対策はしない。

答え・解説 正答：ア

ア：DB状態と送信予定を同一コミットに含め、後段で再試行可能にする考え方が有効である。
イ：注文と通知が不整合になる。
ウ：永続性と整合性を失う。
エ：重複処理を引き起こす。
総合解説：DB更新とメッセージングの二重書込み問題には、Outbox等で一方の正本に送信予定を確実に記録し再試行できる方式がある。

対象：2026年度 / 基準日 2026-09-01

0086 データ・連携設計 > DB整合性・トランザクション設計 ｜ 難易度：基礎

一般に、DBトランザクション内で利用者の長時間入力待ちを避ける理由として適切なものはどれか。

ア．長時間保持すると索引が不要になるから。
イ．ロックや資源保持時間が長くなり、競合やタイムアウトを増やし得るため。
ウ．トランザクションは長いほど必ず安全だから。
エ．利用者入力はDBが直接受け付ける必要があるから。

答え・解説 正答：イ

ア：索引の必要性とは無関係である。
イ：トランザクションを長くすると同時実行性や資源利用へ悪影響が出る。
ウ：安全性と長さは単純比例しない。
エ：アプリケーション層で入力を受けられる。
総合解説：トランザクション境界は業務整合性を保ちながら必要最小限にし、競合を抑える。

対象：2026年度 / 基準日 2026-09-01

0087 データ・連携設計 > DB整合性・トランザクション設計 ｜ 難易度：基礎

DBMSのトランザクションログを障害回復に利用する主な目的はどれか。
ア．利用者のパスワードを平文で記録する。
イ．索引を必ず不要にする。
ウ．コミット済み更新の再実行や未完了更新の取消しに必要な情報を保持する。
エ．画面レイアウトを保存する。

答え・解説 正答：ウ

ア：認証情報を平文保存するものではない。
イ：索引の要否とは別である。
ウ：更新履歴を利用して整合した状態へ復旧するために用いる。
エ：UI情報とは目的が異なる。
総合解説：障害回復ではログ、チェックポイント、バックアップを組み合わせ、原子性と耐久性を支える。

対象：2026年度 / 基準日 2026-09-01

0088 データ・連携設計 > API・メッセージ・外部システム連携 ｜ 難易度：基礎

外部API仕様において、項目「amount」を定義する際に必要な情報として最も適切なものはどれか。
ア．項目名だけを書き意味は口頭で伝える。
イ．利用者ごとに単位を自由に解釈させる。
ウ．エラー時の扱いは仕様に記載しない。
エ．意味、単位、型、桁数、必須性などを明確にする。

答え・解説 正答：エ

ア：属人的な伝達では一貫性を保てない。
イ：単位不一致は重大な誤処理につながる。
ウ：異常系も契約の一部である。
エ：データの意味と制約を共有しなければ相互運用できない。
総合解説：API契約ではデータモデル、意味、制約、エラー、認証、版などを明確にする。

対象：2026年度 / 基準日 2026-09-01

0089 データ・連携設計 > API・メッセージ・外部システム連携 ｜ 難易度：標準

ネットワーク切断時に安全な再試行を行いたい更新APIで最も重要な設計上の検討はどれか。
ア．同じ要求を複数回受けても結果が重複しないよう冪等性を設計する。
イ．毎回異なる副作用を必ず発生させる。
ウ．応答コードを全て200に固定する。
エ．要求IDを記録しない。

答え・解説 正答：ア

ア：通信結果不明時の再試行で二重処理を防ぐため冪等性が重要である。
イ：重複副作用の危険が増える。
ウ：状態に合った応答を返す設計が必要である。
エ：重複判定や追跡が難しくなる。
総合解説：外部APIは通信失敗を前提に、冪等性、タイムアウト、再試行、重複検知を設計する。

対象：2026年度 / 基準日 2026-09-01

0090 データ・連携設計 > API・メッセージ・外部システム連携 ｜ 難易度：標準

メッセージ基盤が少なくとも1回配送を保証し、同じイベントが重複して届く可能性がある。消費側の設計として適切なものはどれか。
ア．処理結果を記録しない。
イ．イベントID等で処理済みを判定し、重複しても二重更新しないようにする。
ウ．重複は絶対にないと仮定する。
エ．同じイベントを受けるたび売上を加算する。

答え・解説 正答：イ

ア：重複判定や再処理が難しくなる。
イ：重複配送を受けても同じ結果になるよう冪等化する。
ウ：配送保証の前提に反する。
エ：重複による二重計上が起きる。
総合解説：メッセージングでは配送保証の意味を理解し、重複・順序・再試行を利用側設計へ反映する。

対象：2026年度 / 基準日 2026-09-01

**0091** データ・連携設計 > API・メッセージ・外部システム連携 ｜ 難易度：標準

同じ「在庫変更」イベントを、補充システム、分析システム、通知システムがそれぞれ独立に受け取りたい。適切な方式はどれか。

ア．一つの利用者だけが受信できる専用ファイルに固定する。
イ．在庫変更を紙で通知する。
ウ．Publish/Subscribe方式を検討する。
エ．在庫システムのDBを各システムが直接更新する。

**答え・解説** 正答：ウ

ア：複数独立利用者という要件に合わない。
イ：即時自動連携の要件に適さない。
ウ：一つの発行を複数購読者へ配信する用途に適する。
エ：データ所有境界が崩れる。
総合解説：Pub/Subは発行者と複数購読者を疎結合にできるが、スキーマ、再配信、順序等を設計する。

対象：2026年度 / 基準日 2026-09-01

**0092** データ・連携設計 > API・メッセージ・外部システム連携 ｜ 難易度：応用

外部公開APIで互換性のない変更が必要になった。適切な対応はどれか。

ア．既存利用者へ通知せず同じ契約を破壊的に変更する。
イ．利用者が多いほど即日停止する。
ウ．変更履歴を残さない。
エ．新バージョンを用意し、旧版の廃止時期と移行手順を利用者へ示す。

**答え・解説** 正答：エ

ア：突然の破壊的変更は連携障害を招く。
イ：多数利用者ほど影響管理が重要である。
ウ：追跡性が低下する。
エ：移行期間と版を管理することで利用者側の改修を計画できる。
総合解説：外部APIは契約として扱い、互換性、版管理、非推奨化、廃止プロセスを定める。

対象：2026年度 / 基準日 2026-09-01

0093 データ・連携設計 > API・メッセージ・外部システム連携 ｜ 難易度：基礎

繰返し処理に失敗するメッセージを通常キューから分離して調査・再処理するための仕組みとして適切なものはどれか。

ア．デッドレターキュー
イ．ロードバランサ
ウ．DNSキャッシュ
エ．ファイル圧縮

答え・解説 正答：ア

ア：処理不能メッセージを隔離し、通常処理の停滞を防ぎながら調査できる。
イ：通信分散の装置でありメッセージ隔離ではない。
ウ：名前解決キャッシュである。
エ：データサイズ削減の処理である。
総合解説：非同期連携では失敗メッセージの隔離、再試行上限、監視、再処理手順を設計する。

対象：2026年度 / 基準日 2026-09-01

0094 データ・連携設計 > API・メッセージ・外部システム連携 ｜ 難易度：応用

API A→B→Cと呼び出す処理で、障害時に一つの利用者要求を横断追跡したい。適切な設計はどれか。

ア．障害時だけ手作業でソースを書き換える。
イ．要求ごとの相関IDを各サービスへ伝播しログに記録する。
ウ．各サービスが無関係なIDを毎行ランダム生成し関連付けない。
エ．ログ時刻を記録しない。

答え・解説 正答：イ

ア：平常時から観測可能性を組み込むべきである。
イ：共通のトレース識別子で複数サービスのログを関連付けられる。
ウ：関連付けできず横断調査が困難になる。
エ：時系列分析も難しくなる。
総合解説：分散システムでは相関ID、メトリクス、ログ、トレースを設計し、外部連携障害の原因を追跡可能にする。

対象：2026年度 / 基準日 2026-09-01

0095 データ・連携設計 > API・メッセージ・外部システム連携 ｜ 難易度：応用

夜間ファイル連携で100万件を受信する。途中で取込処理が失敗しても安全に再実行したい。適切な設計はどれか。

ア．受信件数やチェックサムを確認しない。
イ．処理完了前に元ファイルを削除する。
ウ．ファイル識別子や処理状態を記録し、重複取込を防ぎつつ再開・再実行できるようにする。
エ．失敗したら毎回全件を無条件追加する。

答え・解説 正答：ウ

ア：完全性を検証できない。
イ：復旧元を失う。
ウ：処理単位と状態を管理すれば再実行時の二重登録を防止できる。
エ：重複データを作る。
総合解説：ファイル連携でも一意識別、完全性検証、再実行、重複防止、保管・削除を設計する。

対象：2026年度 / 基準日 2026-09-01

0096 データ・連携設計 > API・メッセージ・外部システム連携 ｜ 難易度：応用

外部APIが障害中なのに大量リクエストを送り続け、自システムのスレッドも枯渇している。改善策として適切なものはどれか。

ア．タイムアウトを無限にする。
イ．再試行間隔を0秒にする。
ウ．全要求を同時に無制限並列化する。
エ．失敗率が閾値を超えたら一定時間呼出しを遮断し、回復確認後に再開する方式を検討する。

答え・解説 正答：エ

ア：資源枯渇を悪化させる。
イ：再試行嵐を引き起こす。
ウ：依存先と自システム双方を圧迫する。
エ：サーキットブレーカ等で故障依存先への呼出しを一時遮断し障害波及を抑えられる。
総合解説：外部依存障害を前提に、タイムアウト、バックオフ、遮断、フォールバックを組み合わせて障害の連鎖を防ぐ。

対象：2026年度 / 基準日 2026-09-01

0097 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：基礎

オンライン照会機能の性能要件として、設計・試験で最も検証しやすい記述はどれか。
ア．通常負荷時は95パーセンタイルの応答時間を2秒以内とし、同時利用者数1,000人の条件で測定する。
イ．利用者が速いと感じること。
ウ．できるだけ高性能なサーバを使うこと。
エ．性能問題が起きたらその都度増強すること。

答え・解説 正答：ア

ア：負荷条件と測定指標、目標値が明示され、試験で判定できる。
イ：主観的で合否判定ができない。
ウ：手段だけで、必要性能の水準が示されていない。
エ：事後対応方針であり、性能要求の定義になっていない。
総合解説：性能要件は応答時間、スループット、同時利用者数、データ量などを組み合わせ、測定条件と目標値を明確にする。

対象：2026年度 / 基準日 2026-09-01

0098 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：標準

日中平均は毎秒200件だが、キャンペーン開始直後の10分間は毎秒1,200件に達する受注APIがある。容量設計で最も適切な考え方はどれか。
ア．平均200件だけを基準にすれば十分である。
イ．ピーク時の負荷特性と継続時間を把握し、必要容量やスケーリング方式を決める。
ウ．最大値は一時的なので性能試験の対象外にする。
エ．平均値とピーク値を足して常に1,400件として設計する。

答え・解説 正答：イ

ア：平均だけではピーク時に要求を満たせない可能性がある。
イ：実際のボトルネックとなるピーク条件を踏まえ、過不足のない容量を設計できる。
ウ：業務上発生するピークは性能設計・試験で考慮すべきである。
エ：負荷の時間特性を無視した単純加算は合理的でない。
総合解説：容量設計では平均値だけでなくピーク値、継続時間、増加傾向、余裕率、障害時構成を合わせて評価する。

対象：2026年度 / 基準日 2026-09-01

0099 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：標準

画像変換処理は1件数秒かかるが、利用者には受付完了を1秒以内に返せばよく、変換結果は数分以内でよい。性能安定化に有効な方式はどれか。

ア．受付要求ごとに画像変換が完了するまで同期的に待たせる。
イ．全変換処理をWebサーバ1プロセスに固定する。
ウ．受付と画像変換を分離し、ジョブキューに投入してワーカが非同期処理する。
エ．ピーク時だけ受付機能を停止する。

答え・解説 正答：ウ

ア：変換時間が直接受付応答時間となり、ピーク時に悪化しやすい。
イ：処理資源の分離・拡張が難しくなる。
ウ：受付応答と重い処理を分離でき、ワーカ数で処理能力を調整しやすい。
エ：サービス要件を満たすための設計ではなく、可用性を下げる。
総合解説：即時完了が不要な処理では非同期化とキューにより負荷を平準化し、処理資源を独立にスケールさせられる。

対象：2026年度 / 基準日 2026-09-01

0100 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：基礎

1台の処理能力が毎秒180件で、ピーク時に毎秒700件を処理する必要がある。障害時の余裕は別途考えない場合、処理能力だけから求めた最小台数は何台か。

ア．3台×180=540件/秒でよいとし、ピーク700件/秒を満たさない。
イ．5台×180=900件/秒を選ぶが、最小台数ではない。
ウ．7台まで必要と過大に見積もる。
エ．4台×180=720件/秒となるため、700件/秒を満たす最小4台とする。

答え・解説 正答：エ

ア：3台では540件/秒で不足する。
イ：5台でも満たすが最小ではない。
ウ：必要台数を過大に見積もっている。
エ：4台なら720件/秒となり、700件/秒を上回る最小台数である。
総合解説：単純な容量計算では必要スループットを1台当たり能力で割り、切り上げる。実設計では障害時容量や余裕率も追加して検討する。

対象：2026年度 / 基準日 2026-09-01

0101 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：応用

負荷試験でWeb層CPUは30%、アプリ層CPUは40%だが、DB接続プールが常時上限に達し、接続待ちが応答時間の60%を占めている。最初に評価すべき改善はどれか。

ア．DB接続の利用時間、プールサイズ、クエリ特性を調べ、接続待ちの原因を解消する。
イ．WebサーバのCPUだけを2倍にする。
ウ．画面画像を高解像度に変更する。
エ．アプリ層を停止してDBだけで画面を提供する。

答え・解説 正答：ア

ア：測定で支配的な待ち要因がDB接続待ちなので、原因に直接働きかける。
イ：Web層CPUに余裕があり、主原因への効果は限定的である。
ウ：画像解像度はDB接続待ちの主原因ではない。
エ：システム機能を成立させない変更である。
総合解説：性能改善は推測ではなく計測からボトルネックを特定し、最も支配的な待ち要因から検証する。

対象：2026年度 / 基準日 2026-09-01

0102 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：標準

Webアプリを水平スケールさせたいが、利用者セッションを各サーバのローカルメモリだけに保持している。改善策として最も適切なものはどれか。

ア．ロードバランサを外し、利用者にサーバIPを選ばせる。
イ．セッションを共有ストアへ外出しするなど、どのインスタンスでも処理できる構成を検討する。
ウ．サーバごとに異なる画面仕様にする。
エ．水平スケールを行うたびに全利用者をログアウトさせることを標準運用とする。

答え・解説 正答：イ

ア：負荷分散機能を失い、運用性も悪化する。
イ：インスタンス固有状態への依存を減らすと、台数増減や障害切替が容易になる。
ウ：機能不整合を生み、スケーラビリティ改善ではない。
エ：継続利用性を損ない、根本的な状態管理問題を解決しない。
総合解説：水平スケールでは状態の持ち方が重要である。ステートレス化や共有状態管理を用いて任意インスタンスで処理できるようにする。

対象：2026年度 / 基準日 2026-09-01

0103 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：標準

性能試験で平均応答時間は目標内だったが、本番では特定の重い検索が集中すると遅延する。再試験の改善として適切なものはどれか。

ア．軽い検索だけを100%実行して再試験する。
イ．データを1件だけに減らして測定する。
ウ．本番の処理比率、データ量、ピーク集中、重い処理を反映したワークロードで測定する。
エ．応答時間の測定をやめ、CPU型番だけで判定する。

答え・解説 正答：ウ

ア：問題となる処理を除外しており再現性がない。
イ：データ量依存の性能劣化を見逃す。
ウ：実利用の負荷構成を再現することで、本番で生じるボトルネックを検出しやすい。
エ：性能要件の合否を実測できない。
総合解説：性能試験は同時数だけでなく、トランザクション構成、データ量、キャッシュ状態、ピークパターンなど本番条件をモデル化する。

対象：2026年度 / 基準日 2026-09-01

0104 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：基礎

現在1TBのデータが毎年50%ずつ増え、3年間は同一システムを使う予定である。容量設計で適切な対応はどれか。

ア．現在1TBなので将来も1TBとして固定する。
イ．ディスク不足が起きるまで監視をしない。
ウ．データ増加は性能やバックアップ時間に影響しないとみなす。
エ．増加率を基に将来容量を見積もり、拡張方法・上限・増設リードタイムを確認する。

答え・解説 正答：エ

ア：既知の増加傾向を無視している。
イ：事前の容量管理ができず、サービス停止につながり得る。
ウ：データ量増加はI/O、バックアップ、復旧時間などにも影響する。
エ：将来のデータ量と増設条件を先に見積もることで、容量不足を予防できる。
総合解説：容量設計は初期値だけでなく成長率とライフサイクルを考慮し、いつ・どのように拡張するかまで決める。

対象：2026年度 / 基準日 2026-09-01

**0105** 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：応用

CPU使用率が急上昇してから新インスタンスが利用可能になるまで8分かかるサービスで、5分程度の急激な負荷スパイクが頻発する。単純なCPU閾値による事後スケールだけでは遅延する。改善として適切なものはどれか。

ア．起動時間を踏まえた最低台数、予測・スケジュール型スケール、キュー長など複数の先行指標を検討する。
イ．CPU閾値を100%にして限界まで待つ。
ウ．新規インスタンスが起動するまで全要求を破棄する。
エ．オートスケールを停止し、常に1台に固定する。

**答え・解説** 正答：ア

ア：スケール反映の遅延と負荷の立上りを考慮し、先行して容量を確保する設計ができる。
イ：増強開始がさらに遅れ、性能悪化を招く。
ウ：要求処理のSLAを満たさない。
エ：変動負荷への追随性を失う。
総合解説：オートスケールは閾値だけでなく、観測指標、起動時間、クールダウン、最低・最大台数、負荷予測を合わせて設計する。

対象：2026年度 / 基準日 2026-09-01

**0106** 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：基礎

応答時間の中央値が0.8秒、95パーセンタイルが1.8秒、99パーセンタイルが8秒である。評価として最も適切なものはどれか。

ア．中央値が1秒未満なので全要求が1秒未満である。
イ．大多数は速いが、上位1%付近に大きな遅延があり、原因を確認する必要がある。
ウ．99パーセンタイルは最小応答時間を表す。
エ．パーセンタイルは処理件数と無関係なので測定不要である。

**答え・解説** 正答：イ

ア：中央値は50%点であり全件の上限ではない。
イ：中央値だけでは見えない長い尾の遅延を99パーセンタイルが示している。
ウ：99パーセンタイルは99%の要求がその値以下である境界である。
エ：利用者体験やSLA評価に有用な指標である。
総合解説：平均・中央値だけでなく高位パーセンタイルを見ると、一部利用者に集中する遅延を把握しやすい。

対象：2026年度 / 基準日 2026-09-01

**0107** 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：標準

単一DBサーバのCPU増強だけで5年間対応してきたが、機種上限に近づき、今後もデータと負荷が増える。次の設計検討として適切なものはどれか。

ア．上限に達しても同じCPUを追加購入すれば無限に増強できるとみなす。
イ．DBの監視を停止して負荷増加を見えなくする。
ウ．読み書き特性や整合性要件を分析し、読取り分散、パーティショニング、アーキテクチャ変更などを評価する。
エ．全クエリを直列化すれば必ずスループットが上がる。

**答え・解説** 正答：ウ

ア：ハードウェアには上限がある。
イ：問題の可視性を下げるだけで性能は改善しない。
ウ：垂直拡張の上限を超える前に、データ・処理特性に応じた水平的な拡張策を評価する。
エ：直列化は並列性を失い、通常はスループット向上策ではない。
総合解説：スケーラビリティ設計では将来の上限を見据え、垂直・水平拡張のトレードオフを早期に評価する。

対象：2026年度 / 基準日 2026-09-01

**0108** 非機能設計 > 性能・容量・スケーラビリティ ｜ 難易度：標準

夜間バッチを並列化したところ、並列数を増やすほどDB I/O待ちが増え、全体時間が逆に長くなった。適切な対応はどれか。

ア．並列数を無制限に増やす。
イ．I/O待ちは性能に影響しないので無視する。
ウ．全ジョブの開始時刻を完全に同じにする。
エ．ボトルネック資源の利用状況を測定し、最適な並列度や処理分割を決める。

**答え・解説** 正答：エ

ア：競合をさらに悪化させる可能性が高い。
イ：待機時間は処理時間へ直接影響する。
ウ：ピーク負荷を集中させ、競合を強める。
エ：共有資源の競合を考慮して並列度を調整すれば、全体スループットを改善できる。
総合解説：並列化は無条件に速くなるわけではない。CPU、I/O、ロック、外部APIなど共有資源の飽和点を測定して調整する。

対象：2026年度 / 基準日 2026-09-01

0109 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：基礎

障害発生後「30分以内にサービスを再開し、失われるデータは最大5分分まで」とする要求の対応として正しいものはどれか。

ア．サービス再開目標をRTO=30分、許容データ損失をRPO=5分と定義する。
イ．二つの意味を逆にし、RTO=5分、RPO=30分と定義する。
ウ．復旧時間だけを基準にし、RTOもRPOも30分とする。
エ．データ損失許容量だけを基準にし、RTOもRPOも5分とする。

答え・解説 正答：ア

ア：RTOは復旧までの目標時間、RPOはどの時点までデータを戻せればよいかを示す。
イ：二つの指標を逆に解釈している。
ウ：復旧時間と許容データ損失は別の要求である。
エ：同様に二つの値を区別していない。
総合解説：障害対策ではサービス復旧時間とデータ損失許容量を分けて定義し、それぞれを満たすバックアップ・冗長化方式を選ぶ。

対象：2026年度 / 基準日 2026-09-01

0110 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：標準

アプリサーバとDBは二重化されているが、両系統が同じ1台の電源装置に依存している。可用性設計上の評価はどれか。

ア．サーバが二重化されているので電源は評価不要である。
イ．共通電源が単一障害点になり得るため、電源系統も含めて冗長性を評価する。
ウ．DBが二重化されていれば電源障害は発生しない。
エ．電源障害はソフトウェアだけで必ず回避できる。

答え・解説 正答：イ

ア：サーバだけの冗長化では共有電源障害を防げない。
イ：サービス経路全体で共通部品を洗い出す必要がある。
ウ：DB冗長化は電源故障そのものをなくさない。
エ：物理電源喪失をソフトウェアだけでは解決できない。
総合解説：冗長化はサーバ台数だけでなく、電源、ネットワーク、ストレージ、拠点など共通障害要因まで確認する。

対象：2026年度 / 基準日 2026-09-01

0111 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：標準

待機系サーバを用意したが、障害時の切替手順を一度も試していない。最も重要な改善はどれか。
ア．待機系が存在するだけで自動的にRTOを保証できるとみなす。
イ．本番障害が起きるまで待機系を起動しない。
ウ．切替条件、切替手順、戻し手順を定義し、定期的にフェイルオーバ訓練を行う。
エ．切替失敗時の対応を決めず、その場で判断する。

答え・解説 正答：ウ

ア：構成だけでは復旧時間を保証できない。
イ：起動・同期・接続切替の問題を事前に検出できない。
ウ：実際に切り替えられることを検証し、運用手順を成熟させる必要がある。
エ：復旧品質が担当者依存になる。
総合解説：可用性は冗長構成だけでなく、検知、切替、復旧、運用訓練まで含めて実現する。

対象：2026年度 / 基準日 2026-09-01

0112 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：基礎

DBをリアルタイム複製しているが、誤操作で大量データを削除した場合に備える設計として適切なものはどれか。
ア．レプリカがあるのでバックアップは常に不要である。
イ．削除操作も複製されるようにすれば復旧できる。
ウ．誤操作対策として監視ログを削除する。
エ．複製とは別に、世代管理されたバックアップやポイントインタイム復旧手段を用意する。

答え・解説 正答：エ

ア：レプリケーションと履歴保全は目的が異なる。
イ：削除が複製されると両系統から消える可能性がある。
ウ：監査・原因究明を困難にする。
エ：論理的な誤削除はレプリカにも反映され得るため、過去時点へ戻せる保全が必要である。
総合解説：高可用性の複製とデータ保全のバックアップは代替関係ではない。障害種別ごとに復旧手段を設計する。

対象：2026年度 / 基準日 2026-09-01

0113 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：応用

外部決済APIが遅延すると、自システムのスレッドが待ち続けて枯渇し、決済以外の機能も停止する。最も適切な改善方針はどれか。

ア．適切なタイムアウト、呼出し数制限、サーキットブレーカやフォールバックを設計して障害波及を抑える。
イ．外部APIへのタイムアウトを無限にする。
ウ．全機能で同じスレッドプールをさらに小さくする。
エ．外部APIが復旧するまで監視を停止する。

答え・解説 正答：ア

ア：外部障害を早く検知・遮断し、自システム資源の枯渇を防ぎやすい。
イ：待機資源が蓄積し、障害連鎖を悪化させる。
ウ：資源枯渇を早める可能性がある。
エ：監視停止は復旧判断を困難にする。
総合解説：分散システムでは部分障害を前提に、タイムアウト、再試行、遮断、隔離などで障害の波及範囲を限定する。

対象：2026年度 / 基準日 2026-09-01

0114 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：標準

注文登録APIで通信タイムアウト時にクライアントが自動再試行する。二重注文を防ぐために有効な設計はどれか。

ア．再試行ごとに必ず新規注文として登録する。
イ．一意な冪等性キーを受け付け、同一要求の重複実行を識別して一度だけ処理する。
ウ．タイムアウトを発生させないため通信障害を無視する。
エ．注文番号を利用者の記憶だけで管理する。

答え・解説 正答：イ

ア：二重注文を増やす設計である。
イ：再送されても同一業務要求を識別でき、重複副作用を抑制できる。
ウ：通信障害は現実に起こり得るため無視できない。
エ：システムとしての一意性・整合性保証にならない。
総合解説：再試行可能な更新処理では、冪等性、重複検知、処理結果照会などを設計し、部分障害時の二重実行を防ぐ。

対象：2026年度 / 基準日 2026-09-01

0115 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：標準

月30日、24時間サービスで、計画・障害を含む停止時間を月43.2分以内に抑える必要がある。この目標可用性に最も近いものはどれか。

ア．停止割合を1%と読み違え、可用性99%と判断する。
イ．停止許容量を10分の1に厳しく見積もり、可用性99.99%と判断する。
ウ．43,200分の0.1%=43.2分なので、可用性99.9%と判断する。
エ．停止割合を5%とみなす誤りにより、可用性95%と判断する。

答え・解説 正答：ウ

ア：1%停止なら432分であり大きすぎる。
イ：0.01%停止なら約4.32分で、より厳しい。
ウ：30日×24時間=43,200分で、0.1%は43.2分なので99.9%に相当する。
エ：5%停止なら2,160分であり大きく異なる。
総合解説：可用性目標は許容停止時間へ換算すると設計・運用上の意味を確認しやすい。

対象：2026年度 / 基準日 2026-09-01

0116 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：基礎

冗長化されたサービスで障害復旧を早めるため、監視設計として最も適切なものはどれか。

ア．サーバの電源ランプだけを毎月確認する。
イ．利用者から苦情が来るまで監視しない。
ウ．CPU使用率だけを見れば全ての障害を検知できるとみなす。
エ．プロセス生存だけでなく、主要機能の外形監視や依存先、エラー率、遅延も組み合わせて監視する。

答え・解説 正答：エ

ア：頻度・観点ともに不足している。
イ：検知が遅れ、MTTRが長くなる。
ウ：CPU正常でもアプリや依存先が故障する可能性がある。
エ：サービスが実際に機能しているかを複数観点から監視できる。
総合解説：信頼性・可用性を高めるには、障害を早期に検知し、原因切分けと復旧へつなげる監視が必要である。

対象：2026年度 / 基準日 2026-09-01

**0117** 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：応用

主系と待機系を同じ建物・同じ電源系統に置いている。サーバ故障には対応できるが、大規模災害時もサービス継続したい。適切な追加検討はどれか。

ア．災害リスクが独立する別拠点への待機系配置、データ複製、切替手順をRTO/RPOに基づき評価する。
イ．同じラック内に待機系をもう1台追加するだけで全災害に対応できる。
ウ．主系と待機系のバックアップを同じディスクに保存する。
エ．災害時は復旧時間目標を無効にする。

**答え・解説** 正答：ア

ア：拠点全体が失われる相関障害を分離し、事業要求に基づいたDRを設計できる。
イ：同一建物障害には共倒れする。
ウ：共通障害点を増やし、保全にもならない。
エ：定義済み事業要求を障害時だけ無効化するのは不適切である。
総合解説：高可用性では障害の粒度を機器・ラック・拠点まで広げ、同時被災する要因を分離する。

対象：2026年度 / 基準日 2026-09-01

**0118** 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：基礎

サービス可用性を改善する施策として、MTTR短縮が直接意味するものはどれか。

ア．障害が発生する平均間隔を長くすること。
イ．障害発生から復旧までの平均時間を短くすること。
ウ．1要求当たりのCPU時間を短くすること。
エ．開発者数を減らすこと。

**答え・解説** 正答：イ

ア：これはMTBF改善に近い意味である。
イ：MTTRは一般に復旧に要する平均時間を表し、短縮すれば停止時間を減らしやすい。
ウ：性能指標でありMTTRとは別である。
エ：人数削減自体は復旧時間の定義ではない。
総合解説：可用性は故障頻度を下げるだけでなく、検知・切替・復旧を速めて停止時間を短縮することでも改善できる。

対象：2026年度 / 基準日 2026-09-01

0119 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：標準

災害時は在庫照会を継続したいが、分析レポートは数時間停止してもよい。適切な設計はどれか。
ア．全機能を同じ優先度にし、資源不足時は全停止する。
イ．分析を優先し、在庫照会を必ず停止する。
ウ．重要機能と非重要機能を区分し、資源不足時は分析を停止して照会を優先する縮退モードを設計する。
エ．障害時の機能優先度を事前に決めない。

答え・解説 正答：ウ

ア：必要以上に停止範囲を広げる。
イ：業務優先度に反する。
ウ：業務継続上重要な機能を優先することで限られた資源でもサービスを維持できる。
エ：障害時の判断が属人化する。
総合解説：可用性設計では全面継続だけでなく、業務重要度に応じた縮退運転や優先順位付けも有効である。

対象：2026年度 / 基準日 2026-09-01

0120 非機能設計 > 可用性・信頼性・障害対策 ｜ 難易度：標準

バックアップジョブは毎日成功ログを出しているが、過去1年間リストアを試していない。評価として適切なものはどれか。
ア．成功ログがあるので復元可能性は100%保証される。
イ．復元テストは本番障害時に初めて行えばよい。
ウ．バックアップ媒体の暗号化だけ確認すれば復元性は不要である。
エ．バックアップの成功だけでなく、定期的に復元テストを行い、RTO/RPOとデータ完全性を確認する。

答え・解説 正答：エ

ア：媒体破損、設定誤り、手順不備などは成功ログだけでは分からない。
イ：本番時の初検証は復旧遅延のリスクが高い。
ウ：暗号化と復元可能性は別の品質である。
エ：実際に復元できることを検証して初めて復旧手段として信頼できる。
総合解説：障害対策は「取得できた」だけでなく「必要時間内に正しく戻せる」ことを定期的に確認する。

対象：2026年度 / 基準日 2026-09-01

0121 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：基礎

バッチ処理用サービスアカウントの権限設計として最も適切なものはどれか。
ア．必要なテーブルと操作だけに権限を限定し、不要な管理者権限を付与しない。
イ．運用を簡単にするため全DBの管理者権限を付与する。
ウ．複数システムで同一の管理者IDを共有する。
エ．権限設定を行わずアプリ側の画面表示だけで制御する。

答え・解説 正答：ア

ア：業務に必要な最小限の権限に限定すると侵害時の影響を抑えられる。
イ：過剰権限により被害範囲が広がる。
ウ：追跡性と分離が低下する。
エ：DB側の実アクセスを制御できない。
総合解説：セキュリティ設計では最小権限、職務分離、認証情報管理をシステム境界ごとに具体化する。

対象：2026年度 / 基準日 2026-09-01

0122 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：標準

アプリケーションが外部APIの秘密鍵を利用する。保守性とセキュリティの両面で適切な管理はどれか。
ア．秘密鍵をソースコードへ直接記述して公開リポジトリに保存する。
イ．ソースコードから分離した秘密管理基盤に保管し、アクセス制御とローテーションを行う。
ウ．全環境で同じ秘密鍵を無期限に使う。
エ．秘密鍵をログへ毎回出力する。

答え・解説 正答：イ

ア：漏えいリスクが非常に高い。
イ：コードと秘密を分離し、権限・更新・監査を管理しやすい。
ウ：環境分離と鍵更新ができず、侵害時の影響が拡大する。
エ：ログから秘密が漏えいする。
総合解説：秘密情報は設定値の一種でも機密性が高い。コードとは分離し、保管、配布、ローテーション、監査を設計する。

対象：2026年度 / 基準日 2026-09-01

0123 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：標準

監査ログに利用者の操作履歴を残す。最も適切な設計方針はどれか。
ア．パスワードを平文で全て記録する。
イ．障害調査のため個人情報を無制限に全項目記録する。
ウ．追跡に必要な主体・時刻・操作・結果を記録しつつ、不要なパスワードや秘密値は記録しない。
エ．ログ改ざん検知やアクセス制御は不要とする。

答え・解説 正答：ウ

ア：認証情報漏えいにつながる。
イ：目的外・過剰な記録はリスクを増やす。
ウ：監査に必要な情報を確保しながら、機密情報の過剰記録を避けられる。
エ：ログ自体の信頼性と機密性を守る必要がある。
総合解説：ログは運用・監査・障害分析に重要だが、記録項目、保管期間、改ざん防止、閲覧権限、機密情報除外を設計する。

対象：2026年度 / 基準日 2026-09-01

0124 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：基礎

毎週の設定変更を担当者が手作業で実施し、入力ミスが頻発している。改善として適切なものはどれか。
ア．担当者ごとに独自手順を増やす。
イ．実行記録を残さないことで作業を短縮する。
ウ．本番環境で直接試行錯誤することを標準にする。
エ．変更手順をコード化・自動化し、レビューと実行記録を残す。

答え・解説 正答：エ

ア：ばらつきが増え、属人化が進む。
イ：監査・原因追跡が困難になる。
ウ：本番事故のリスクが高い。
エ：手順の再現性を高め、レビュー可能にし、操作ミスを減らしやすい。
総合解説：運用性向上では、人手手順を標準化・自動化し、承認、実行、ロールバック、記録を再現可能にする。

対象：2026年度 / 基準日 2026-09-01

0125 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：標準

一つの巨大モジュールに業務ルール、DBアクセス、画面処理が混在し、変更のたびに広範囲な回帰が必要である。改善として適切なものはどれか。

ア．責務を分離し、明確なインタフェースと単体テスト可能な境界を設ける。
イ．さらに全機能を同じモジュールへ追加する。
ウ．DBアクセスを各画面へコピーして分散させる。
エ．テストを削除して変更時間を短縮する。

答え・解説 正答：ア

ア：責務分離により変更影響を局所化し、検証しやすくなる。
イ：結合と変更影響がさらに増える。
ウ：重複コードが増え一貫性を失いやすい。
エ：短期的に見えても品質リスクが増える。
総合解説：保守性はモジュール性、理解容易性、変更容易性、テスト容易性などを設計段階から考慮する。

対象：2026年度 / 基準日 2026-09-01

0126 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：応用

利用中のOSSライブラリに重大な脆弱性が公表されたが、システム内の利用箇所が把握できず対応判断に時間がかかった。再発防止として有効な設計・運用はどれか。

ア．依存ライブラリの版情報を記録しない。
イ．依存コンポーネントと版を継続的に把握し、脆弱性情報と照合して更新・回避策を適用できるプロセスを整備する。
ウ．脆弱性情報はリリース後に一切確認しない。
エ．更新を避けるため全ライブラリを永久に同一版で固定する。

答え・解説 正答：イ

ア：影響範囲を特定できなくなる。
イ：利用資産の把握と脆弱性監視を結び付けることで、影響評価と対応を迅速化できる。
ウ：運用期間中の新たな脆弱性に対応できない。
エ：既知脆弱性を抱え続ける可能性がある。
総合解説：長期運用システムでは、構成情報、依存関係、脆弱性情報、更新試験、ロールバックを一連の保守プロセスとして設計する。

対象：2026年度 / 基準日 2026-09-01

0127 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：標準

24時間サービスで月1回のOS更新が必要である。停止を極小化する方式として適切なものはどれか。
ア．全系統を同時に停止してから手順を考える。
イ．更新を永久に実施しない。
ウ．冗長構成の片系ずつ更新し、正常性確認後に切り替えるローリング方式を要件に応じて採用する。
エ．本番中に検証なしで全ノードを一斉更新する。

答え・解説 正答：ウ

ア：停止時間が大きくなり、事前設計も不足している。
イ：セキュリティ・保守性のリスクが増える。
ウ：サービスを維持しながら段階的に更新でき、問題時の影響を限定しやすい。
エ：一斉障害のリスクが高い。
総合解説：保守性と可用性は独立ではない。更新方式、冗長度、互換性、切戻しを合わせて設計する。

対象：2026年度 / 基準日 2026-09-01

0128 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：基礎

分散システムで障害原因の追跡に時間がかかる。設計段階で有効な対応はどれか。
ア．各サービスの時計設定をばらばらにする。
イ．本番ログを全て無効化する。
ウ．エラーが起きても正常応答として記録する。
エ．ログ、メトリクス、トレースに共通の相関IDを持たせ、要求経路を追跡できるようにする。

答え・解説 正答：エ

ア：時刻の不整合で時系列解析が難しくなる。
イ：調査材料を失う。
ウ：障害検知を困難にする。
エ：複数サービスを跨ぐ処理を関連付け、原因切分けを容易にできる。
総合解説：運用性には監視だけでなく、障害時に内部状態を把握し、原因を特定できる観測可能性の設計も含まれる。

対象：2026年度 / 基準日 2026-09-01

0129 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：応用

新しい設定を本番反映した直後にエラー率が上昇した。安全な変更設計として事前に用意すべきものはどれか。
ア．変更前状態の保存、段階適用、監視基準、切戻し条件と手順を定める。
イ．変更前設定を削除し、戻せないようにする。
ウ．エラー率の監視を停止する。
エ．全環境へ同時適用し、結果確認は翌月に行う。

答え・解説 正答：ア

ア：異常を早期検知し、影響を限定して元へ戻せる。
イ：復旧可能性を失う。
ウ：変更失敗を検知できない。
エ：影響範囲が最大化し、検知も遅い。
総合解説：変更容易性と運用安全性を両立するには、変更の可逆性、段階展開、観測、承認を設計する。

対象：2026年度 / 基準日 2026-09-01

0130 非機能設計 > セキュリティ・運用性・保守性 ｜ 難易度：標準

社内ネットワーク内からの通信は全て信頼し、認証を省略する設計について最も適切な評価はどれか。
ア．社内IPなら利用者本人であることが必ず保証される。
イ．内部侵害や端末乗っ取りも考慮し、重要操作では通信元だけでなく主体認証・権限確認を行うべきである。
ウ．内部通信にはアクセス制御が不要である。
エ．一度VPN接続すれば全システムの管理者権限を与えるべきである。

答え・解説 正答：イ

ア：IPアドレスは本人性を保証しない。
イ：ネットワーク位置だけを信頼根拠にせず、主体と権限を確認する方がリスクを抑えられる。
ウ：内部にも不正・侵害リスクがある。
エ：過剰権限となり最小権限に反する。
総合解説：セキュリティ設計では境界の内外を問わず、資産価値とリスクに応じて認証・認可・監査を組み合わせる。

対象：2026年度 / 基準日 2026-09-01

0131 設計評価・文書化 > アーキテクチャ評価 ｜ 難易度：標準

「障害に強いこと」という要求をアーキテクチャ評価へ使える形にするため、最も適切な方法はどれか。
ア．「とにかく高信頼」とだけ記載する。
イ．構成図の見た目だけで評価する。
ウ．障害の種類、発生条件、許容停止時間、復旧方法などを具体的な評価シナリオにする。
エ．製品価格が高いほど信頼性も十分と判断する。

答え・解説 正答：ウ

ア：判定基準がなく評価できない。
イ：図の美観は品質達成の証拠ではない。
ウ：具体的な刺激・環境・期待結果を定めると方式が要求を満たすか検証しやすい。
エ：価格だけでは要求適合を判断できない。
総合解説：アーキテクチャ評価は抽象的な品質語ではなく、性能・可用性・保守性などを具体的なシナリオと基準にして行う。

対象：2026年度 / 基準日 2026-09-01

0132 設計評価・文書化 > アーキテクチャ評価 ｜ 難易度：基礎

二つのアーキテクチャ案を比較するとき、最も適切な評価方法はどれか。
ア．設計者が最初に思いついた案を採用する。
イ．製品名の知名度だけで決める。
ウ．全ての評価項目を無視し、初期費用だけで決める。
エ．要件ごとの評価基準を定め、性能、可用性、コスト、運用性、移行リスクなどを同じ前提で比較する。

答え・解説 正答：エ

ア：代替案評価が行われていない。
イ：知名度は要件適合を保証しない。
ウ：初期費用以外の重要品質を無視する。
エ：複数の重要要求と制約に対して比較することで判断根拠が明確になる。
総合解説：アーキテクチャ選定では単一指標ではなく、要求優先度とトレードオフを明示して代替案を比較する。

対象：2026年度 / 基準日 2026-09-01

0133 設計評価・文書化 > アーキテクチャ評価 ｜ 難易度：応用

採用候補の新しいストリーム処理基盤が、要求する毎秒5万件を安定処理できるか実績がない。設計確定前の対応として適切なものはどれか。

ア．本番相当データと負荷で技術検証（PoC）を行い、処理能力・遅延・障害時挙動を測定する。
イ．カタログの最大値だけを根拠に確定する。
ウ．性能リスクを文書から削除する。
エ．本番導入後に初めて負荷試験する。

答え・解説 正答：ア

ア：不確実性が大きい高リスク項目を早期に実測し、方式判断へ反映できる。
イ：条件差を確認できず、要求達成の証拠にならない。
ウ：リスク自体は消えない。
エ：問題発見が遅く手戻りが大きい。
総合解説：アーキテクチャ評価では、未知の技術や重要な性能仮定などを早い段階で検証し、設計リスクを低減する。

対象：2026年度 / 基準日 2026-09-01

0134 設計評価・文書化 > アーキテクチャ評価 ｜ 難易度：標準

当初は同時利用者1,000人を前提に設計したが、事業計画変更で10,000人へ増えた。適切な対応はどれか。

ア．設計は一度承認されたので前提変更を無視する。
イ．性能・容量・構成など、当初前提に依存したアーキテクチャ判断を再評価する。
ウ．利用者数は非機能設計に影響しないとみなす。
エ．要件文書だけ更新し、設計は確認しない。

答え・解説 正答：イ

ア：旧前提での適合性しか保証されない。
イ：重要前提が変われば、関連する設計判断の妥当性も変わり得る。
ウ：容量・スケーラビリティへ直接影響する。
エ：設計との不整合を残す。
総合解説：アーキテクチャ決定には前提と制約を紐付け、前提変更時に影響する判断を追跡・再評価できるようにする。

対象：2026年度 / 基準日 2026-09-01

0135 設計評価・文書化 > アーキテクチャ評価 ｜ 難易度：応用

レビューで次の二つが見つかった。A: 低頻度だが発生時に全取引を24時間停止する単一障害点。B: 高頻度だが画面表示が0.1秒遅くなる軽微な問題。最初に詳細評価すべきものはどれか。

ア．必ず頻度だけでBを優先する。
イ．両方とも評価せず実装後に考える。
ウ．影響が極めて大きいAを、発生可能性と事業影響を踏まえて優先評価する。
エ．画面の色を変更してリスクを相殺する。

答え・解説 正答：ウ

ア：重大な事業停止リスクを見落とす。
イ：早期設計段階でのリスク低減機会を失う。
ウ：リスクは発生可能性だけでなく影響度を合わせて優先順位付けする。
エ：リスク原因と無関係である。
総合解説：設計評価では要求未達のリスクを発生可能性と影響の両面から評価し、高いものから検証・対策する。

対象：2026年度 / 基準日 2026-09-01

0136 設計評価・文書化 > アーキテクチャ評価 ｜ 難易度：基礎

重要なアーキテクチャを評価する際、設計担当者だけでなく運用・セキュリティ・業務担当も参加させる主な理由はどれか。

ア．設計者の責任をなくすため。
イ．参加人数が多いほど自動的に正しい設計になるため。
ウ．文書化を不要にするため。
エ．異なる品質要求や運用制約から設計の見落としを発見しやすくするため。

答え・解説 正答：エ

ア：設計責任や意思決定は別途明確にする必要がある。
イ：人数そのものは品質を保証しない。
ウ：共通理解のため文書はむしろ重要である。
エ：多様な関係者は性能、運用、セキュリティ、業務継続など異なる観点を提供できる。
総合解説：アーキテクチャは複数の品質特性と運用に影響するため、関係者の観点を取り入れて評価する。

対象：2026年度 / 基準日 2026-09-01

**0137** 設計評価・文書化 > アーキテクチャ評価 ｜ 難易度：標準

性能は案Aが優れるが、運用コストと復旧容易性では案Bが優れ、最終的にBを採用した。文書化として適切なものはどれか。

ア．評価基準、測定結果、重み付け・優先順位、採用理由と残存リスクを記録する。
イ．採用案Bの名前だけ記録し、比較結果は破棄する。
ウ．不採用案Aが存在したことを隠す。
エ．全ての品質特性が同等だったことに書き換える。

**答え・解説** 正答：ア

ア：将来の変更時に判断根拠とトレードオフを再確認できる。
イ：なぜBを選んだか追跡できない。
ウ：意思決定の透明性を失う。
エ：事実と異なる記録になり、再評価を妨げる。
総合解説：評価結果は単なる点数ではなく、意思決定の根拠、前提、残存リスクとして設計文書へ残す。

対象：2026年度 / 基準日 2026-09-01

**0138** 設計評価・文書化 > 設計レビュー・要件トレーサビリティ ｜ 難易度：基礎

設計レビューで、性能要件NFR-07に対応する設計要素と性能試験項目がトレーサビリティ表上で空欄になっている。最も適切な対応はどれか。

ア．要件変更を禁止する。
イ．NFR-07を実現する設計要素と検証する試験項目を特定し、対応関係を明示した上で要件未実現がないか確認する。
ウ．設計文書を全て一つのファイルにする。
エ．全要件に同じ優先度を付ける。

**答え・解説** 正答：イ

ア：要件変更はあり得るため、禁止するのではなく影響を追跡できることが重要である。
イ：空欄は要件が設計・検証へ割り当てられていない可能性を示すため、実現先と検証先を確認する。
ウ：文書を一つにまとめても、要件と設計・試験の対応関係は自動的には明確にならない。
エ：優先順位を一律にしても、未実現要件の検出にはつながらない。
総合解説：トレーサビリティ表の空欄は、要求が設計や検証へ落ちていない可能性を示す。設計レビューでは要求ごとの実現先と検証先をたどり、漏れを解消する。

対象：2026年度 / 基準日 2026-09-01

0139 設計評価・文書化 > 設計レビュー・要件トレーサビリティ ｜ 難易度：標準

可用性要件が「月間停止43分以内」である。設計レビューで最も直接確認すべきものはどれか。
ア．画面の配色が統一されているかだけを確認する。
イ．設計書のページ数が多いか確認する。
ウ．冗長構成、障害検知・切替時間、計画停止を含めて停止時間目標を満たせる根拠があるか。
エ．サーバ製品の価格が最も高いか確認する。

答え・解説 正答：ウ

ア：可用性要件への直接的な確認にならない。
イ：文書量は要求達成を示さない。
ウ：要件と設計メカニズム・定量根拠を対応させて適合性を確認できる。
エ：価格は可用性達成の根拠ではない。
総合解説：設計レビューでは、要件ごとに設計がどう満たすか、定量条件や異常系まで含めて確認する。

対象：2026年度 / 基準日 2026-09-01

0140 設計評価・文書化 > 設計レビュー・要件トレーサビリティ ｜ 難易度：応用

「取引履歴を1年保存」から「7年保存」へ要件変更された。適切な影響分析はどれか。
ア．要件文の数字だけ7へ変更し、設計は触らない。
イ．保存期間は画面だけの要件なのでDBへ影響しない。
ウ．変更履歴を削除し、旧要件との違いを残さない。
エ．関連するDB容量、バックアップ、検索性能、アーカイブ、個人情報管理、テストを追跡し再評価する。

答え・解説 正答：エ

ア：設計と要件の不整合を残す。
イ：容量・運用・性能へ直接影響する。
ウ：変更理由と影響の追跡性を失う。
エ：保存期間変更は複数の非機能・運用設計へ波及するため、追跡関係から影響を確認する。
総合解説：要求変更は起点だけでなく、関連設計・テスト・運用へ連鎖する。トレーサビリティを用いて漏れなく影響分析する。

対象：2026年度 / 基準日 2026-09-01

0141 設計評価・文書化 > 設計レビュー・要件トレーサビリティ ｜ 難易度：標準

設計レビューで多数の指摘が出た。指摘管理として適切なものはどれか。
ア．指摘内容、根拠、重大度、担当、対応期限、対応結果を記録し、クローズ条件を明確にする。
イ．指摘件数だけを数え、内容を残さない。
ウ．設計者が不快に感じる指摘は記録しない。
エ．全指摘を同じ優先度で即日対応する。

答え・解説 正答：ア

ア：対応漏れを防ぎ、重要な問題から計画的に解消できる。
イ：何を直すべきか追跡できない。
ウ：品質上必要な問題を隠すことになる。
エ：重大度や影響に応じた優先順位付けができない。
総合解説：レビューは指摘を出すことが目的ではなく、問題を追跡して解決し、設計品質を確認するまでが一連の活動である。

対象：2026年度 / 基準日 2026-09-01

0142 設計評価・文書化 > 設計レビュー・要件トレーサビリティ ｜ 難易度：基礎

設計要素から元の要件へ逆方向にも追跡できることの利点はどれか。
ア．要件書を削除できる。
イ．どの要件にも対応しない不要な設計や、根拠不明の機能を発見しやすい。
ウ．全ての設計変更を自動承認できる。
エ．テストを実施しなくてよくなる。

答え・解説 正答：イ

ア：根拠資料は引き続き必要である。
イ：設計の存在理由を確認でき、過剰実装やスコープ逸脱を検出しやすい。
ウ：承認プロセスの代替ではない。
エ：追跡性はテストそのものを代替しない。
総合解説：要件→設計だけでなく設計→要件も追跡できると、実現漏れと過剰実装の双方を点検できる。

対象：2026年度 / 基準日 2026-09-01

**0143** 設計評価・文書化 > 設計レビュー・要件トレーサビリティ ｜ 難易度：標準

大規模設計を全て完成させてから一度だけレビューしたところ、基本方式の誤りで大幅な手戻りが発生した。改善として適切なものはどれか。

ア．レビューを廃止して実装者に任せる。
イ．全設計を実装後まで非公開にする。
ウ．重要な方式・境界・非機能設計など節目ごとに段階レビューし、早期に重大問題を検出する。
エ．誤りが見つからないようレビュー観点を減らす。

**答え・解説** 正答：ウ

ア：品質確認機会を失う。
イ：問題発見がさらに遅れる。
ウ：高影響の判断を早期に確認し、後工程の手戻りを抑えられる。
エ：品質低下につながる。
総合解説：レビューは成果物の成熟度とリスクに応じて適切なタイミングで行い、重大な設計誤りを早く除去する。

対象：2026年度 / 基準日 2026-09-01

**0144** 設計評価・文書化 > 設計レビュー・要件トレーサビリティ ｜ 難易度：標準

構成図ではDBが二重化されているが、復旧手順書では単一DBを前提に手順が記載されている。レビューでの対応として適切なものはどれか。

ア．構成図と手順書は別文書なので矛盾していてよい。
イ．どちらが正しいか決めず両方を残す。
ウ．手順書だけを削除して問題を隠す。
エ．文書間の前提不一致を指摘し、正しい設計へ統一して関連文書を更新する。

**答え・解説** 正答：エ

ア：同じシステムの運用に関わるため整合している必要がある。
イ：実運用で判断できず事故要因になる。
ウ：必要な運用情報を失うだけである。
エ：設計・運用文書の整合性を保ち、障害時の誤操作を防げる。
総合解説：設計レビューでは単一文書内だけでなく、要件、構成、インタフェース、運用手順など成果物間の整合性も確認する。

対象：2026年度 / 基準日 2026-09-01

**0145** 設計評価・文書化 > UML・モデル・設計文書 ｜ 難易度：基礎

複数オブジェクト間で、ある処理が時間順にどのメッセージをやり取りするか表現したい。最も適したUML図はどれか。

ア．シーケンス図
イ．クラス図
ウ．配置図
エ．パッケージ図

**答え・解説** 正答：ア

ア：ライフライン間のメッセージを時系列で表現するために適している。
イ：静的な型・属性・関係を表す用途が中心である。
ウ：実行環境のノードと配置物を表す用途が中心である。
エ：モデル要素のグルーピングや依存を表す用途が中心である。
総合解説：UMLでは目的に応じて図を使い分ける。相互作用の時系列はシーケンス図が適している。

対象：2026年度 / 基準日 2026-09-01

**0146** 設計評価・文書化 > UML・モデル・設計文書 ｜ 難易度：標準

注文が「受付」「支払待ち」「出荷待ち」「完了」「取消」の状態を持ち、イベントに応じて遷移する。振る舞いを表すのに適したUML図はどれか。

ア．配置図
イ．状態機械図
ウ．コンポーネント図
エ．オブジェクト図

**答え・解説** 正答：イ

ア：ノードへの配置を表す図である。
イ：状態とイベントによる遷移を表現する目的に適している。
ウ：コンポーネントと提供・要求インタフェース等を表す。
エ：ある時点のオブジェクト構造を表す。
総合解説：状態に応じて許される操作や遷移が変わる対象では、状態機械図によりライフサイクルを明確にできる。

対象：2026年度 / 基準日 2026-09-01

**0147** 設計評価・文書化 > UML・モデル・設計文書 ｜ 難易度：標準

Webサーバ、アプリサーバ、DBサーバという実行ノードと、それぞれに配置される成果物の関係を表現したい。最も適したUML図はどれか。

ア．ユースケース図
イ．アクティビティ図
ウ．配置図
エ．クラス図

**答え・解説** 正答：ウ

ア：利用者とシステム機能の関係を表す用途が中心である。
イ：処理フローや制御・データの流れを表す用途が中心である。
ウ：実行環境のノードと配置される成果物・通信経路を表す用途に適している。
エ：静的な型構造を表す用途が中心である。
総合解説：物理・実行環境の構成を設計文書へ表す場合、UML配置図が有効である。

対象：2026年度 / 基準日 2026-09-01

**0148** 設計評価・文書化 > UML・モデル・設計文書 ｜ 難易度：応用

クラス図で「顧客」側1に対し「注文」側0..*の関連がある。この意味として最も適切なものはどれか。

ア．一つの顧客には必ず注文が1件だけ存在する。
イ．顧客と注文の間に関連は存在しない。
ウ．一つの注文に顧客が必ず0人だけ対応する。
エ．一つの顧客に注文が0件以上対応し、各注文はこの関連上で一つの顧客に対応する。

**答え・解説** 正答：エ

ア：注文側の多重度0..*と矛盾する。
イ：関連線があるため無関係ではない。
ウ：1側の意味を誤っている。
エ：0..*は0個以上、1はちょうど1個を表し、この関連の個数制約を示す。
総合解説：UMLクラス図の多重度は関連端ごとの個数制約を表す。モデルを業務ルールやデータ設計と整合させることが重要である。

対象：2026年度 / 基準日 2026-09-01

**0149** 設計評価・文書化 > UML・モデル・設計文書 ｜ 難易度：標準

受注後に「請求書作成」と「出荷指示」を並行して開始し、両方完了後に完了通知する処理をモデル化したい。適切な表現はどれか。

ア．アクティビティ図で分岐ではなくforkで並行化し、joinで同期してから完了通知へ進む。
イ．クラス図の継承関係だけで表現する。
ウ．配置図でサーバ台数だけを増やす。
エ．ユースケース名を長くして並行性を文章に埋め込むだけにする。

**答え・解説** 正答：ア

ア：UMLアクティビティ図ではfork/joinを用いて並行フローと同期を表現できる。
イ：静的型関係では処理順序・並行性を直接表せない。
ウ：物理配置の図では業務フローの同期を表現しにくい。
エ：モデル上の制御構造が明確にならない。
総合解説：モデルは伝えたい設計情報に合う図を選ぶ。並行する業務・処理フローにはアクティビティ図が有効である。

対象：2026年度 / 基準日 2026-09-01

**0150** 設計評価・文書化 > UML・モデル・設計文書 ｜ 難易度：応用

同じサービスを、概要構成図では「決済サービス」、詳細設計では「PaymentCore」、運用手順では「請求サーバ」と呼び、対応関係の説明がない。最も適切な改善はどれか。

ア．文書ごとに自由な名称を増やし、説明は口頭だけにする。
イ．用語・識別子の対応を定義し、各モデルの抽象度と対象範囲を明示して文書間を相互参照できるようにする。
ウ．概要図を削除し、詳細コードだけを唯一の設計文書にする。
エ．運用担当には設計文書を見せない。

**答え・解説** 正答：イ

ア：名称不一致がさらに増え、属人化する。
イ：異なる抽象度のモデル間でも同一要素を追跡でき、関係者の誤解を減らせる。
ウ：関係者ごとに必要な抽象度が異なるため、詳細コードだけでは十分でない。
エ：運用設計との整合確認を妨げる。
総合解説：設計文書は図を増やすこと自体が目的ではない。用語、識別子、対象範囲、版、相互参照を管理し、複数の視点で同じシステムを一貫して説明できることが重要である。

対象：2026年度 / 基準日 2026-09-01