Engineer

Architecture

comment les parties d'un système s'articulent et pourquoi

Ce que c’est

L’architecture, souvent appelée architecture logicielle ou système en informatique, est l’ensemble des concepts et propriétés fondamentaux d’un système dans son environnement, portés par ses éléments, leurs relations et les principes gouvernant sa conception et son évolution. Cette formulation suit la définition de NIST SP 800-160 Volume 1 Revision 1, Engineering Trustworthy Secure Systems, publié en novembre 2022.

Chaque partie de la définition a un rôle. Les éléments sont les pièces : services, bases de données, files d’attente, interface et modules internes. Les relations décrivent leurs dépendances et ce qui passe entre elles. Les principes décident de l’emplacement des nouveaux travaux et empêchent éléments et relations de dériver pendant la croissance. « Dans son environnement » inclut utilisateurs, systèmes voisins et contraintes dans l’architecture, et « évolution » signifie qu’une description vraie seulement le jour de son dessin est incomplète.

Tout choix de conception n’est pas architectural. Le groupe qui maintient le format Architectural Decision Record (ADR) définit une décision architecturale comme « un choix de conception justifié qui répond à une exigence fonctionnelle ou non fonctionnelle significative sur le plan architectural ». Une exigence est significative lorsque son effet sur l’architecture et la qualité peut être mesuré. Le modèle arc42 est plus direct : sa section décisions vise les « décisions importantes, coûteuses, risquées ou controversées ».

Comme aucun dessin ne montre tout, l’architecture se décrit par vues. Le modèle C4 de Simon Brown propose des diagrammes comparables à des cartes où l’on change d’échelle : contexte situant le système parmi utilisateurs et voisins, conteneurs montrant applications et stockages internes, composants d’un

conteneur puis, lorsque cela aide, code. arc42 organise la description écrite en douze sections, des objectifs et contraintes aux décisions, risques et dette technique.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que l'architecture ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Définir d'abord le système sur le papier pour que ses éléments s'articulent et que son coût soit connu avant d'écrire du code.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Un plan d'infrastructure pour 14 départements — 14 départements de recherche sur un même plan d'infrastructure

Un plan d'infrastructure pour 14 départements Contactez-nous →

Pourquoi c’est important

La décision que l’architecture place devant une équipe porte moins sur la structure à choisir que sur les choix à faire maintenant, délibérément et par écrit, et ceux à laisser ouverts.

Trancher tôt une question structurelle fait payer avant d’en savoir beaucoup : le choix se fait avec le plus faible niveau d’information dont le projet disposera au cours de son existence et ralentit les premières semaines. La laisser ouverte fixe malgré tout la structure, une modification pratique après l’autre, par la dernière personne ayant touché le code. La sortie consiste à séparer deux types de choix. Quelques-uns deviennent coûteux à modifier dès que le code en dépend : découpage du système, propriétaire des règles métier, accès aux données, niveau exigé de chaque changement. Ils méritent d’être décidés tôt et volontairement. La plupart des autres restent faciles à revoir et peuvent attendre que le code montre ses besoins.

La consignation est la seconde moitié de la décision, celle où survit le raisonnement. Un ADR conserve une décision, sa justification, ses compromis et ses conséquences ; leur collection forme le journal des décisions du projet. Le site ADR attribue la popularisation à l’article de Michael Nygard de 2011, « Documenting Architecture Decisions ». Le coût est une page écrite par décision. Le bénéfice est qu’un lecteur ultérieur peut distinguer compromis délibéré et accident, puis vérifier si la raison tient encore.

Les objectifs de qualité départagent des structures qui fonctionneraient toutes. arc42 demande les trois principaux, cinq au maximum, ceux qui importent le plus aux parties prenantes majeures, et avertit qu'ils sont distincts des objectifs du projet. Deux conceptions aux mêmes fonctions peuvent différer entièrement dans leurs pannes, leur montée en charge ou leur évolution. Seuls les objectifs de qualité disent quelles différences comptent.

À quoi cela ressemble en pratique

Le dépôt de la plateforme s'est ouvert vide le 25 novembre 2025. La page sur ses décisions fondatrices explique que ces choix sont « peu coûteux à faire au troisième jour et presque impossibles à inverser au sixième mois ». Deux des trois décisions d'alors ont façonné la structure. Chaque couche a une tâche : interface Next.js, API Go validant et transmettant, fonctions PostgreSQL propriétaires des règles métier. Les accès sont passés dès le départ par des fonctions stockées, car l'ajouter après coup « aurait impliqué de réécrire chaque gestionnaire ». Neuf mois et plusieurs milliers de commits plus tard, selon la page, les couches restent distinctes et aucun gestionnaire n'interroge directement une table. Le test d'une décision fondatrice n'est pas sa plausibilité initiale, mais sa résistance au volume de travail ultérieur.

Une structure laissée au hasard se fixe par accident, comme le montre la page sur le développement de cartes SIG. La carte avait grandi par morceaux : données spatiales dans PostgreSQL, dessin dans Mapbox, ReactJS en façade et NodeJS derrière. Chaque partie fonctionnait seule, mais elles « n'avaient tout simplement pas été conçues pour s'assembler ». Le rôle couvrait le développement de bout en bout et la direction technique, et l'architecture a été définie d'abord. Ensuite, les responsabilités « étaient régulièrement ramenées à leur couche d'origine dès que l'une commençait à déborder sur la suivante », une grande part du travail consistant à « repérer les petites incohérences avant qu'elles ne se figent en architecture ».

La page du plan de gestion des conteneurs traite la raison écrite comme le véritable résultat. Le moteur habituel de l'industrie publie les ports des conteneurs en amont des règles du pare-feu de l'hôte. Le plan de gestion utilise donc un moteur sans démon pilotant des unités générées par le superviseur du système, sans aucun port publié. Un compromis est explicitement consigné : les unités fonctionnent au niveau système plutôt qu'en mode sans privilèges root, ce qui simplifie l'automatisation au prix de l'isolation plus stricte de l'autre mode. La note indique dans quel sens évoluer si la protection contre les évasions de conteneurs devient plus importante. C'est ce qu'un ADR conserve : décision,

justification, compromis et conséquences. La page place l'écriture de la raison au-dessus du choix lui-même, car « tous les articles que lira la personne suivante lui conseilleront le choix conventionnel ».

Où les problèmes surviennent

Une nouvelle personne consciencieuse inverse un choix délibéré. Sa raison n'existait que dans une tête : le choix ressemble donc à une erreur, et le choix conventionnel recommandé par tous les tutoriels à une correction. Écrivez un ADR pour chaque décision coûteuse à inverser : contexte, options envisagées, choix, conséquences et critères justifiant un réexamen. Conservez ces fiches à côté du code, pour qu'une modification de l'un soit examinée avec l'autre.

Les règles métier apparaissent simultanément dans trois couches. Chaque modification a placé sa logique là où c'était pratique cet après-midi-là, jusqu'à vérifier la même règle dans l'interface, l'API et la base, un peu différemment partout. Nommez les responsabilités de chaque couche, écrivez-les et rapatriez-les dès qu'elles débordent, pendant que la copie parasite est petite. Si la plateforme le permet, imposez aussi la frontière, par exemple en autorisant l'API à appeler des fonctions tout en lui refusant les droits sur les tables.

Une règle introduite au sixième mois devient un projet de nettoyage. La décision dont tout dépend a été laissée ouverte, et chaque commit a supposé l'état du moment. Décidez d'abord les rares éléments coûteux et laissez volontairement le reste ouvert. Dites-le dans le journal des décisions : une question reste alors visiblement ouverte plutôt que discrètement tranchée par le prochain changement.

Le diagramme d'architecture n'explique rien à personne. Un dessin mélange utilisateurs, services, classes et tables, avec des lignes sans étiquette ou portant « utilise ». Faites un diagramme par échelle comme C4, et donnez à chaque ligne la direction et l'intention de la relation. Entre conteneurs, nommez aussi technologie ou protocole. Les conseils de notation C4 demandent précisément cela et déconseillent les étiquettes d'un mot comme « Utilise ».

Pour aller plus loin

- Architecture, glossaire NIST CSRC — définition utilisée ici, à côté des définitions FIPS et d'architecture de sécurité de la même entrée.

- NIST SP 800-160 Vol. 1 Rev. 1 — Engineering Trustworthy Secure Systems, publication d'origine de la définition.
- Architectural Decision Records — critères d'une décision architecturale, modèles de consignation et outils utiles.
- Le modèle C4 — quatre échelles de diagrammes, du contexte au code, avec conseils de notation et liste de revue.
- arc42 — conseils pratiques et exemples pour chacune des douze sections du modèle arc42.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que l'architecture ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/architecture/