Engineer

архітектура

🧩 як частини системи поєднуються між собою і чому

Що це таке

Архітектура, яку в програмуванні часто називають архітектурою програмного забезпечення або архітектурою системи, — це сукупність фундаментальних понять і властивостей системи в її середовищі, втілених у її елементах, у зв'язках між ними та в принципах, що визначають, як систему проєктують і як вона змінюється. Це формулювання йде за визначенням із NIST SP 800-160 Volume 1 Revision 1, Engineering Trustworthy Secure Systems, опублікованого в листопаді 2022 року.

Кожна частина визначення має своє призначення. Елементи — це складники: сервіси, бази даних, черги, фронтенд, модулі всередині них. Зв'язки — це те, як складники залежать один від одного і що між ними передається. Принципи — це правила, які вирішують, куди лягає нова робота, і не дають елементам і зв'язкам розповзатися, коли система росте. Слова «в її середовищі» роблять частиною архітектури користувачів, сусідні системи й обмеження, а те, що визначення говорить і про розвиток у часі, означає: опис, правдивий лише в день, коли його накреслили, неповний.

Не кожне проєктне рішення архітектурне. Група, що підтримує формат Architectural Decision Record (ADR), називає архітектурним рішенням обґрунтований проєктний вибір, який відповідає на функціональну чи нефункціональну вимогу, значущу для архітектури, а вимога значуща, коли має вимірний вплив на архітектуру та якість системи. Шаблон arc42 каже прямолінійніше: його розділ про рішення призначено для важливих, дорогих, ризикових або суперечливих рішень.

Одне креслення ніколи не показує всього, тому архітектуру описують у кількох видах. Модель C4, яку створив Simon Brown, пропонує діаграми, що працюють як карти, які можна наближати й віддаляти: діаграму системного

контексту, що показує, де система стоїть серед своїх користувачів і систем навколо, діаграму контейнерів із застосунками та сховищами даних усередині неї, діаграму компонентів для одного контейнера і, де це допомагає, діаграму коду. arc42 упорядковує письмовий опис у дванадцять розділів — від цілей і обмежень до рішень, ризиків і технічного боргу.

Застосуйте поняття до рішення

Застосуйте «Що таке архітектура?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Структура системи, спершу продумана на папері, щоб частини пасували одна до одної, а вартість була відома до першого рядка коду.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Один інфраструктурний план для 14 підрозділів — 14 дослідницьких підрозділів, охоплених спільним інфраструктурним планом

Один інфраструктурний план для 14 підрозділів Зв'язатися з нами →

Чому це важливо

Рішення, яке архітектура ставить перед командою, стосується не так вибору структури, як того, які рішення ухвалити зараз, свідомо й письмово, а які лишити відкритими.

Якщо розв’язати структурне питання рано, платити доведеться ще до того, як багато відомо: вибір роблять, маючи найменше знань, які в проєкту колись будуть, і він сповільнює перші тижні. Якщо лишити його відкритим, структура все одно складеться — одна зручна зміна за раз, руками того, хто останнім торкався коду. Вихід у тому, щоб розрізняти два види рішень. Небагато з них дорого змінювати, щойно від них залежить код: як система ділиться на частини, якій частині належать бізнес-правила, як здійснюється доступ до даних, яку планку має долати кожна зміна. Їх варто ухвалювати рано й навмисно. Більшість решти дешево переглянути, і вони можуть почекати, доки код покаже, чого він потребує.

Запис — друга половина рішення, і саме в ньому виживає обґрунтування. ADR фіксує одне рішення з його обґрунтуванням, компромісами й наслідками, а сукупність ADR проєкту утворює його журнал рішень. Сайт ADR приписує популяризацію ідеї допису Michael Nygard «Documenting Architecture Decisions» 2011 року. Ціна — сторінка тексту на кожне рішення. Віддача — пізніший читач може відрізнити свідомий компроміс від випадковості й перевірити, чи причина досі в силі.

Цілі якості вирішують між структурами, які всі працювали б. arc42 просить назвати три найважливіші, щонайбільше п'ять, цілі якості для архітектури — ті, що найбільше важать для основних зацікавлених сторін, — і застерігає, що це не те саме, що цілі проєкту. Два дизайни з однаковими функціями можуть зовсім по-різному відмовляти, масштабуватися чи змінюватися, і лише цілі якості кажуть, яка різниця має значення.

Як це виглядає на практиці

Репозиторій платформи з'явився 25 листопада 2025 року зовсім порожнім, і сторінка про її засадничі рішення стверджує, що такі рішення «дешево ухвалити на третій день і майже неможливо переграти на шостому місяці». Два з трьох рішень, ухвалених тоді, задали структуру. Стек поділено на рівні так, щоб кожна частина мала одне завдання: фронтенд на Next.js, Go API, який перевіряє дані й передає їх далі, і рівень функцій PostgreSQL, якому належать бізнес-правила. Доступ до даних від самого початку сховано за збереженими функціями, бо «запровадити його згодом означало б переписати кожен обробник». Через дев'ять місяців і кілька тисяч комітів, за словами сторінки, рівні не розмилися й жоден обробник не звертається до таблиці напряму. Перевірка засадничого рішення для неї — не те, чи звучало воно слушно, а те, чи витримало воно обсяг роботи, що надійшов потім.

Структура, залишена на волю випадку, складається випадково, як показує сторінка про GIS-карту. Карта розросталася частинами: просторові дані в PostgreSQL, малювання в Mapbox, ReactJS на фронтенді та NodeJS на бекенді. Кожна частина працювала сама по собі, але «їх не будували так, щоб вони пасували одна до одної». Роль охоплювала full-stack-розробку і технічне керівництво, і спочатку було визначено архітектуру. Далі обов'язки «щоразу повертали на той рівень, де їм місце, щойно якийсь із них починав просочуватися в сусідній», а значна частина роботи полягала в тому, щоб «вчасно помічати дрібні неузгодженості, перш ніж вони встигали затвердіти в архітектурі».

Сторінка про контейнерне середовище вважає справжнім результатом записану причину. Галузевий стандарт серед контейнерних рушіїв публікує порти контейнерів із вищим пріоритетом, ніж правила брандмауера хоста, тож це середовище — контейнерний рушій без демона, що керує юнітами, згенерованими власним супервізором операційної системи, і зовсім без опублікованих портів. Один компроміс записано саме як компроміс: юніти працюють на системному рівні, а не в rootless-режимі, що лишає автоматизацію нескладною ціною суворішої ізоляції, яку дав би rootless-

режим, а в примітці сказано, у який бік рухатися, якщо втеча з контейнера колись важитиме більше. Саме це й містить ADR: рішення з його обґрунтуванням, компромісами й наслідками. Сторінка ставить запис вибору разом із причиною вище за сам вибір, бо «наступній людині кожна стаття, яку вона прочитає, радитиме взяти стандартний».

Де виникають проблеми

Уважний новачок скасовує рішення, яке було свідомим. Причина жила лише в чиїйсь голові, тож вибір схожий на помилку, а стандартний варіант, який радить кожен посібник, схожий на виправлення. Пишіть ADR для кожного рішення, яке дорого скасувати: контекст, зважені варіанти, обраний, його наслідки і те, що зробило б його вартим перегляду. Тримайте записи поруч із кодом, щоб зміну в одному переглядали разом з іншим.

Бізнес-правила з’являються одразу на трьох рівнях. Кожна зміна клала свою логіку туди, де того дня було зручно, аж поки те саме правило не почали перевіряти у фронтенді, в API і в базі даних, щоразу трохи інакше. Назвіть, що належить кожному рівню, запишіть це й повертайте обов’язок на місце, щойно він просочується, поки зайва копія ще мала. Де платформа дозволяє, закріпіть межу ще й технічно: наприклад, дозвольте API викликати функції бази даних, але не давайте йому прав на таблиці.

Правило, запроваджене на шостому місяці, перетворюється на проєкт із прибирання. Рішення, від якого залежить усе, лишили відкритим, і кожен коміт відтоді спирався на те, що випадково було правдою. Спершу ухваліть кілька дорогих рішень, а решту навмисно лишіть відкритими. Напишіть про це в журналі рішень, щоб відкрите питання було видимо відкритим, а не тихо розв’язаним наступною зміною.

Архітектурна діаграма нікому нічого не пояснює. Одне креслення змішує користувачів, сервіси, класи й таблиці, а лінії між блоками або не мають підпису, або підписані одним словом «Uses». Малюйте по одній діаграмі на кожен рівень масштабу, як це робить C4, і підписуйте кожну лінію відповідно до напрямку й призначення зв’язку; між контейнерами називайте ще й технологію чи протокол. Настанови C4 щодо нотації вимагають саме цього й радять уникати підписів з одного слова.

Що почитати далі

- architecture, NIST CSRC Glossary — визначення, використане вище, поряд із визначенням FIPS і визначенням архітектури безпеки, які ділять ту саму статтю.
- NIST SP 800-160 Vol. 1 Rev. 1 — Engineering Trustworthy Secure Systems, публікація, з якої взято визначення.
- Architectural Decision Records — що робить рішення архітектурним, шаблони для його запису й інструменти, які в цьому допомагають.
- The C4 model — чотири рівні діаграм, від системного контексту до коду, з порадами щодо нотації та чеклистом для рев'ю.
- arc42 — практичні поради й приклади до кожного з дванадцяти розділів шаблону arc42.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке архітектура?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/architecture/