Engineer

UI

частина програми, яку людина бачить і з якою працює

Що це таке

UI — скорочення від user interface, інтерфейс користувача: екрани, меню, поля, кнопки, повідомлення й звуки, через які людина каже програмі, що робити, і дізнається, що та зробила. До інтерфейсу належать і правила, що стоять за ними: який елемент керування досяжний звідки, що відбувається після клацання і що програма каже, коли щось іде не так.

Найширше визначення дає глосарій NIST: інтерфейс користувача — це «the physical or logical means by which users interact with a system, device or process», тобто фізичні або логічні засоби, якими користувачі взаємодіють із системою, пристроєм чи процесом. Фізичний бік — клавіатура, сенсорний екран, ручка на приладі. Логічний бік — усе, що малюється чи виводиться у відповідь: вікно з меню, рядок, що чекає на команду, голос, який просить назвати номер. Графічний інтерфейс, командний рядок і голосове меню — усе це інтерфейси користувача, і одна програма часто має кілька з них.

У вебі будівельний блок називає WCAG. *User interface component* — це «a part of the content that is perceived by users as a single control for a distinct function»: частина вмісту, яку користувач сприймає як один елемент керування для окремої функції. Примітки до визначення рахують компонент за тим, що користувач бачить як один елемент керування, а не за тим, як його запрограмовано. Тому один програмний елемент може містити кілька компонентів. Приклад самого стандарту — елемент, що переміщує вмістом по рядку, по сторінці або в довільну точку, де кожен спосіб переміщення потребує власної назви і вважається окремим компонентом. Елементи форм, посилання й елементи керування, створені скриптами, теж рахуються.

Поруч ідуть два суміжні поняття. *UX*, досвід користувача, — ширше питання про те, яким є користування продуктом від початку до кінця: чи виконано завдання, скільки часу воно забрало і як обробляються помилки. Інтерфейс — та частина цього досвіду, яку видно на екрані й на яку можна вказати пальцем. *Дизайн-система* — спільний набір компонентів, токенів і правил, з якого команда будує інтерфейс, щоб два екрани, зроблені двома людьми у два різні місяці, і далі виглядали й поводилися як один продукт.

Застосуйте поняття до рішення

Застосуйте «Що таке UI?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Інтерфейси, що швидко завантажуються, працюють на телефоні й поводяться однаково в усіх браузерах ваших клієнтів.

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. Керувала full-stack-розробкою GIS-карт — full-stack-картографія — від бази даних до браузера

Керувала full-stack-розробкою GIS-карт Зв'язатися з нами →

Чому це важливо

Цінність програми доходить до людей через інтерфейс або не доходить зовсім. Функція, яку може знайти лише той, хто вже знає, де вона, для більшості користувачів так і не вийшла. Такий розрив рідко видно в наборі тестів, бо кожна функція працює. Його видно у зверненнях до підтримки й у формах, кинутих на півдорозі.

Послідовність вивчають один раз і використовують знову. Люди переносять вивчене на одному екрані на наступний. Критерій успіху 3.2.4 у WCAG 2.2, Consistent Identification, вимагає, щоб компоненти з однаковою функцією в межах набору вебсторінок позначалися однаково. Документ Understanding прямо пояснює чому: коли та сама функція в двох місцях має різні назви, люди можуть вирішити, що це дві різні дії, або витратити час, щоб з’ясувати, що вони однакові.

Помилки — частина інтерфейсу. Кожна форма стикається з неправильним введенням, і те, як вона відповідає, — таке саме дизайнерське рішення, як її макет. Критерій 3.3.1, Error Identification, вимагає, щоб виявлена помилка вказувала на поле й описувала проблему текстом. Критерій 3.3.3, Error Suggestion, іде далі: коли програма знає, як виправити автоматично виявлені помилкові вхідні дані, вона пропонує виправлення, якщо це не загрожує

безпеці або призначенню вмісту. Приклад з документа Understanding — поле, яке очікує назву місяця: людині, що ввела «12», пропонують або перелік допустимих значень, або запитання, чи мався на увазі грудень.

Кожен елемент керування треба пояснити й програмам. Критерій 4.1.2, Name, Role, Value, вимагає, щоб для кожного компонента інтерфейсу, зокрема створеного скриптами, програми могли визначити назву й роль і щоб зміни стану до них доходили. Елемент, який лише має вигляд елемента керування на екрані, невидимий для допоміжних технологій, на які покладається багато людей.

Як це виглядає на практиці

Найпоказовіший приклад — морська платформа, де UI та UX продукту спроєктовано від початку до кінця. Вона показує фахівців, компанії, судна, вакансії й відгуки і мала бути досить щільною, щоб умістити справжні морські дані й не перетворитися на стіну. Її тримають кілька рішень. Сітки каталогів перемикаються між картками й таблицею, і застосунок пам’ятає обраний вигляд. Форми показують вимоги над полем введення як живий список, де кожне правило стає зеленим, коли його виконано, і жовтогарячим, поки ні, тож людей спрямовують під час введення, а не виправляють після надсилання. Хлібні крихти й двоколонковий макет повторюються на кожній сторінці сутності, і окремі екрани читаються як один продукт. Помилки кудись ведуть: переспрямування замість глухого кута. Сторінка сама наводить аргумент довіри: на платформі, якою люди користуються заради кар’єри, недбалий вигляд змусив би їх менше довіряти даним, і вони мали б рацію.

Комерційний бік найкраще видно у 200 покращеннях UI/UX для картографічного застосунку GIS. Інтерфейс став потужним і складним, і добра функціональність ховалася за незграбною взаємодією. Замість здогадок роботу почали з відгуків і даних про використання, склали беклог із 200 конкретних змін, ранжували їх за впливом і провели найважливіші через команду, аж поки вони не вийшли як справжні функції. Інтерфейсом стало помітно легше користуватися, а робота сприяла зростанню доходу в 10 разів.

Утримати інтерфейс послідовним із часом — окреме завдання, і його розв’язано в дизайн-системі у стилі «Liquid Glass» з iOS 26. Візуальна мова — напівпрозорі панелі з розмиттям фону, багатошарові тіні, трохи заломлення світла й пружинні анімації, зібрані з утиліт Tailwind. Кожен екран складається

з фіксованого набору компонентів, серед них Card, Label, Button, GlassIconButton і DirectoryGrid. Правила лінтера відхиляють саморобний бейдж чи чип і вказують на канонічний компонент, а документація поєднує конкретні помилки з їхніми виправленнями. Послідовність перестає залежати від того, хто сьогодні переглядає зміну, і її тримають інструменти.

Де виникають проблеми

- **Колір сам несе значення.** Поле із зеленою рамкою для правильного введення й червоною для неправильного нічого не каже тому, хто не розрізняє цих кольорів. Документ Understanding для Use of Color називає саме цей випадок і вимагає другої візуальної ознаки. Слово, значок чи зміна форми поруч із кольором розв'язують проблему.
- **Намальоване як елемент керування ним не є.** Стилізований блок з обробником клацання має вигляд кнопки для того, хто користується мишею, а для того, хто працює з клавіатурою чи екранним читачем, його немає. Власні елементи керування платформи мають роль, фокус і поведінку клавіатури, які імітації доводиться відтворювати вручну, частина за частиною.
- **Кожен екран винаходить свої компоненти заново.** Кожен новий вигляд отримує власні кнопки, бейджі й картки, трохи відмінні одне від одного, аж поки одна зміна не означає правки п'яти версій того самого. Один набір компонентів, який тримає лінтер, а не пам'ять, не дає їм розходитися.
- **Помилки приходять пізно й далеко від причини.** Форма, що приймає все, а потім показує вгорі червоний блок з усім, що було не так, змушує людей шукати поля. Коли поле названо, проблему описано словами поруч із ним і запропоновано виправлення, та сама перевірка стає підказкою.
- **Інтерфейс збудовано під стіл того, хто його робив.** Великий монітор, миша, гострий зір і швидке з'єднання описують автора, а не тих, хто користуватиметься. Телефон в одній руці, збільшена сторінка чи клавіатура без миші — кожне з цього показує проблеми, яких за робочим столом не видно.

Що почитати далі

- NIST CSRC Glossary: user interface — однорядкове визначення з NIST SP 800-152, яке охоплює і фізичні, і логічні засоби взаємодії із системою.
- WCAG 2.2: user interface component — визначення через сприйнятий елемент керування, з примітками про те, як компоненти співвідносяться з елементами, що їх реалізують.

- Understanding SC 3.2.4: Consistent Identification — чому компоненти з однаковою функцією мають мати однакову назву на всьому сайті і для яких повторюваних компонентів це найважливіше.
- Understanding SC 3.3.3: Error Suggestion — що має сказати форма, коли знає, як виправити неправильне введення, з розібраними прикладами.
- Understanding SC 1.4.1: Use of Color — коли колір може нести значення, а коли потрібна друга візуальна ознака.

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 UI?» 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/ui/