Варфрейм вебінтерфейсу: **робочий процес від сценарію до передачі в розробку**

Практичний порядок дій для команди, яка хоче погодити логіку інтерфейсу до візуального дизайну

У ЦЬОМУ ГАЙДІ

Варфрейм потрібен не для того, щоб намалювати сірі прямокутники, а щоб до початку дизайну зафіксувати шлях користувача, склад екранів і наслідки кожної дії. Робота починається з оцінки ризику, а завершується передачею однієї зрозумілої версії логіки дизайнеру та розробці.

---

01 · 1. ВИЗНАЧТЕ, ЧИ ПОТРІБНО ОКРЕМО ПРОЄКТУВАТИ СТРУКТУРУ

Не кожна зміна потребує варфрейму. Перед стартом потрібно відрізнити локальне завдання з очевидною логікою від ситуації, де помилка в структурі змінить користувацький сценарій і призведе до дорогих правок уже під час дизайну або розробки.

Оцініть не візуальну складність екрана, а ризик неправильно зрозуміти дію користувача. Заміна тексту, іконки, візуального акценту або перестановка елемента на сторінці зі зрозумілою логікою зазвичай не вимагають окремого каркаса. Для таких змін достатньо точного коментаря або швидкого ескізу.

Варфрейм потрібен, коли команда ще не може однозначно відповісти, які блоки має бачити користувач, у якому порядку вони з'являються, що запускає кнопка та що стається після дії. Це типова ситуація для нового продукту, каталогу з фільтрами, особистого кабінету, багатокрокової форми, сценаріїв із ролями та різними правами доступу.

Оберіть формат роботи пропорційно до ризику. Повний варфрейм фіксує ієрархію, навігацію, переходи, стани та взаємодії. Швидкий ескіз допомагає погодити головні блоки й базовий шлях. Комбінований підхід доречний, коли складними є лише ключові сценарії, а інші екрани повторюють знайомі патерни.

01 **Сформулюйте цільову дію**
Запишіть, що саме має завершити користувач: знайти товар, заповнити форму, перейти до розділу або змінити дані. Якщо ціль дії нечітка, неможливо перевірити структуру екрана.

02 **Перевірте, чи змінюється сценарій**
Поставте запитання: чи вплине зміна розташування, послідовності блоків або логіки елемента на шлях користувача? Якщо так, робіть варфрейм хоча б для цього сценарію.

03 **Позначте точки невизначеності**
Випишіть місця, де учасники по-різному уявляють екран, наступний крок, склад форми або поведінку кнопки. Саме ці питання має зняти структура до візуального оформлення.

04 **Оберіть рівень робочого матеріалу**
Для локальної зміни використайте коментар або ескіз. Для складного сценарію створіть варфрейм. Якщо складна лише частина продукту, деталізуйте ключові потоки, а повторювані екрани ведіть у макет.

> ДЕ ВТРАЧАЮТЬ
> Найчастіша втрата часу — почати обговорювати колір кнопки або стиль картки, коли команда ще не погодила, куди ця кнопка веде і що має відбутися після натискання.

ЧЕКЛІСТ РОЗДІЛУ

- [ ] Зафіксуйте одну цільову дію для кожного сценарію.

  Без цілі екран перетворюється на перелік блоків, а не на інструмент для виконання задачі.

- [ ] Відокремте локальні правки від змін, що впливають на шлях користувача.

  Без цього команда або витратить зайвий час на деталізацію, або перенесе структурні проблеми в дизайн і розробку.

- [ ] Визначте, які частини продукту потребують повного варфрейму, а які — лише ескізу.

  Однакова деталізація для всіх екранів створює зайву роботу там, де сценарій уже очевидний.

02 · 2. ЗБЕРІТЬ МАРШРУТ КОРИСТУВАЧА ДО ЗАВЕРШЕНОЇ ДІЇ

Каркас інтерфейсу стає корисним тоді, коли показує не окремі екрани, а послідовність дій між ними. Завдання команди — побудувати маршрут, на якому видно звичайний сценарій, повернення, помилки, порожні дані та успішне завершення.

Починайте не з набору екранів, а з маршруту: що користувач бачить спочатку, яку дію робить далі, що змінюється після цієї дії та де сценарій завершується. Для каталогу це може бути шлях від перегляду товарів до застосування фільтра і повернення з порожньої видачі. Для форми — введення даних, помилка, виправлення та успішне надсилання.

Окремо розгляньте ролі користувачів. Якщо різні ролі бачать різний контент або мають різні права, це мають бути явні розгалуження в структурі. Не залишайте такі правила у пам'яті команди чи в усних домовленостях: роль може відкривати, приховувати або робити недоступним цілий розділ чи окрему дію.

Для кожної дії фіксуйте наслідок. Кнопка не є декоративним елементом: вона або відкриває форму, або надсилає дані, або веде на інший екран, або змінює доступний стан. Якщо наслідок не описаний, дизайнер і розробка можуть однаково правильно прочитати схему по-різному.

01 **Намалюйте основний сценарій від старту до результату**
Побудуйте найкоротший маршрут до цільової дії. Назвіть кожен екран і з'єднайте їх переходами, щоб було видно, що саме запускає наступний крок.

02 **Додайте звичайний стан кожного екрана**
Позначте доступні дані, контент, навігацію та дії, які користувач може виконати за нормального перебігу сценарію.

03 **Додайте відсутні дані та помилки**
Для списків покажіть порожній стан. Для форм покажіть помилку заповнення або надсилання. Для кожного такого стану вкажіть, як користувач продовжує сценарій.

04 **Позначте успішне завершення**
Зафіксуйте, що користувач бачить після успішної дії: повідомлення, оновлений екран, наступний крок або повернення до попереднього етапу.

05 **Перевірте залежності між виборами**
Якщо вибір у формі змінює наступне поле чи доступну дію, покажіть цю залежність окремо. Не припускайте, що її можна зрозуміти лише з розташування блоків.

ДЕ ВТРАЧАЮТЬ

Команда часто погоджує лише «ідеальний» шлях. У результаті порожня видача, помилка поля або відсутність доступу з’являються вже в розробці, коли рішення доводиться приймати без узгодженої логіки.

ЧЕКЛІСТ РОЗДІЛУ

- [ ] Для кожного ключового сценарію покажіть старт, дію, результат і спосіб повернення.
  Без повернення користувач може опинитися в тупиковому стані, який не був помічений на основному маршруті.
- [ ] Позначте порожні стани, помилки та успішне завершення.
  Без цих станів розробка отримає лише частину поведінки інтерфейсу.
- [ ] Винесіть ролі та права доступу в окремі правила або гілки сценарію.
  Без цього різний доступ користувачів буде інтерпретовано вже під час реалізації.
- [ ] Перевірте, чи має кожна кнопка конкретну дію та наслідок.
  Без цього однакова область на схемі може стати посиланням, кнопкою або карткою залежно від виконавця.

Варфрейм має бути настільки точним, наскільки дорого помилитися в логіці. Завдання не в тому, щоб зробити сірий макет схожим на готовий дизайн, а в тому, щоб зняти неоднозначність щодо структури, дій, станів і правил взаємодії.

Низька деталізація підходить для зрозумілих сценаріїв без залежних дій. На такій схемі достатньо показати основні блоки, порядок інформації, головний контент і місце, де починається дія користувача. Вона відповідає на питання «що є на сторінці» та «що тут головне».

Середня деталізація потрібна, коли важливі назви полів, навігація, логіка кнопок і базові стани. На цьому рівні команда має бачити, які дані вводяться, що відкриває кожна дія, куди веде посилання та що з'являється після взаємодії.

Висока деталізація потрібна для особистих кабінетів, складних форм, різних ролей і сценаріїв із правилами. Тут фіксують валідацію полів, залежності між виборами, повідомлення системи, права доступу та крайні сценарії. Водночас кольори, тіні, типографіка, ілюстрації та декоративні рішення не повинні відволікати від перевірки логіки.

01 **Оцініть вартість структурної помилки**
Визначте, що станеться, якщо команда по-різному зрозуміє екран: користувач не завершить дію, дані не надішлються, роль отримає зайвий доступ або сценарій зупиниться. Чим суттєвіший наслідок, тим вищою має бути деталізація.

02 **Зафіксуйте склад і порядок блоків**
Для кожного екрана позначте, які блоки є, що головне, де починається дія та в якій послідовності користувач отримує інформацію.

03 **Опишіть поля, кнопки та навігацію**
Додайте назви полів, призначення елементів і напрям переходів. Для кнопки вкажіть не лише текст, а й наслідок натискання.

04 **Опишіть правила взаємодії у ризикових місцях**
Для залежних форм, ролей і змінних даних зафіксуйте умови появи, приховування або блокування елементів, а також повідомлення про помилки й успіх.

05 **Приберіть декоративну деталізацію**
Залиште лише те, що допомагає прочитати ієрархію та сценарій. Якщо обговорення переходить до кольорів, шрифтів і тіней, поверніть команду до поведінки інтерфейсу.

> ДЕ ВТРАЧАЮТЬ
>
> Надмірна деталізація на нестабільних вимогах витрачає час, але недостатня деталізація складної форми переносить критичні питання туди, де їх найдорожче змінювати — у дизайн і розробку.

ЧЕКЛІСТ РОЗДІЛУ

- [ ] Для кожного екрана визначте рівень деталізації за ризиком, а не за його візуальною складністю.

  Без цього прості екрани можуть бути перевантажені, а складні — залишитися без необхідних правил.

- [ ] Перевірте, що всі залежні поля, права доступу та системні повідомлення описані окремо.

  Без окремого опису ці правила легко загубити між екранами.

- [ ] Залиште у варфреймі лише елементи, потрібні для перевірки структури й поведінки.

  Візуальні деталі зміщують фокус із логіки на суб'єктивне обговорення стилю.

04 · 4. ЗАФІКСУЙТЕ РІШЕННЯ В ЖУРНАЛІ ПЕРЕВІРКИ ТА ПРОВЕДІТЬ ПРИЙМАННЯ

Схема стає спільним робочим документом лише після перевірки. Потрібно не просто показати екрани учасникам, а пройти кожен ключовий сценарій, зафіксувати рішення, відкриті питання та критерії, за якими структура вважається погодженою.

Проведіть перевірку як послідовне проходження сценарію. Учасник має побачити початковий екран, виконати дію, отримати результат, зіткнутися з помилкою або порожнім станом і повернутися до роботи. Такий підхід виявляє проблеми, які не видно на наборі статичних екранів.

Розподіліть ролі до обговорення. Представник бізнесу підтверджує ціль і обмеження. Дизайнер перевіряє, чи всі екрани та стани можна оформити. Розробник перевіряє, чи зрозумілі дії, переходи, поля, залежності та права доступу. Відповідальний за контент визначає, які дані мають змінюватися через CMS.

Ведіть журнал перевірки для кожного сценарію. Це прибирає ситуацію, коли рішення існує лише в чаті або пам'яті учасників. Журнал не має бути складним: важливо пов'язати конкретний екран чи елемент із дією, очікуваним результатом, станом, відповідальним і рішенням.

01 **Підготуйте журнал перевірки**
Створіть таблицю з полями: сценарій, екран або елемент, дія користувача, очікуваний результат, стан, відкрите питання, рішення, відповідальний і статус погодження.

02 **Призначте відповідальних за рішення**
До сесії визначте, хто підтверджує бізнес-ціль, хто відповідає за дизайн-наслідки, хто перевіряє реалізацію та хто визначає змінний контент.

03 **Пройдіть сценарій у звичайному стані**
Перевірте, чи видно потрібну дію, чи зрозуміло її призначення та чи потрапляє користувач на очікуваний наступний екран.

04 **Пройдіть виняткові стани**
Окремо перевірте порожні дані, помилку в полі, помилку надсилання, успішне завершення, повернення до попереднього кроку та обмеження для ролі.

05 **Прийміть структуру за критеріями**
Вважайте сценарій готовим лише тоді, коли для нього є названі екрани, зрозумілі переходи, визначені стани, правила доступу за потреби та рішення для всіх критичних відкритих питань.

ДЕ ВТРАЧАЮТЬ

Фраза «все зрозуміло» без проходження сценарію не є погодженням. Вона не фіксує, як має поводитися форма, що відбувається без даних і хто ухвалив рішення щодо спірного елемента.

ЧЕКЛІСТ РОЗДІЛУ

- [ ] Створіть журнал перевірки з рішенням для кожного відкритого питання.
  Без журналу домовленості губляться між погодженням, дизайном і розробкою.
- [ ] Пройдіть ключові сценарії в нормальному, порожньому, помилковому та успішному станах.
  Без цього команда перевірить лише частину реальної поведінки інтерфейсу.
- [ ] Призначте відповідального за бізнес-ціль, дизайн, реалізацію та контентні дані.
  Без розподілу ролей рішення можуть залишитися без власника.
- [ ] Погодьте критерії приймання до переходу у візуальний дизайн.
  Без критеріїв структура може вважатися готовою для одних учасників і неповною для інших.

Після погодження варфрейм потрібно передати як опис поведінки, а не як набір картинок. Кожен екран, блок і дія мають мати контекст: для чого вони існують, що змінюють, за якими умовами з'являються та які дані повинні бути доступними для редагування.

Почніть передачу з карти екранів і зв'язків між ними. Назва екрана має пояснювати його призначення, а стрілки або примітки — показувати напрям переходу. Так дизайнер бачить обсяг станів, які потрібно оформити, а розробка отримує послідовність поведінки, а не лише композицію сторінок.

Додайте контекст до конкретних елементів. Для блоку вкажіть його призначення. Для кнопки — дію та результат. Для поля — правило введення й реакцію на помилку. Для навігації — куди вона веде. Для умовних елементів — коли вони з'являються, зникають або стають недоступними.

Окремо позначте контент, який має змінюватися через CMS. Загальна фраза «зробити як на схемі» не пояснює, чи текст статичний, чи керований редактором, чи картка є посиланням, чи кнопка надсилає форму. Передача має розділяти питання логіки та питання візуального стилю: спершу команда розуміє, що користувач бачить і робить, потім визначає, як це виглядатиме.

01 **Підготуйте карту екранів**
Дайте кожному екрану назву та призначення. Покажіть зв'язки між екранами, включно з переходами після успішної дії, помилки або повернення.

02 **Додайте специфікацію елементів**
Для полів, кнопок, блоків і навігації вкажіть призначення, дію, наслідок і пов'язані стани. Коментарі прив'язуйте до конкретного елемента, а не до екрана загалом.

03 **Позначте правила даних і CMS**
Вкажіть, який контент редактор має змінювати через CMS, а який є частиною незмінної структури. За потреби додайте умови появи або недоступності контенту.

04 **Передайте список обов'язкових станів**
Окремо перелічіть нормальні, порожні, помилкові та успішні стани, а також правила для ролей і залежних дій. Це не має залишатися лише на схемах окремих екранів.

05 **Розділіть логічне та візуальне погодження**
Підтвердіть, що структура, сценарії та правила вже погоджені. Після цього обговорюйте кольори, шрифти, сітку, зображення й деталі компонентів як окремий етап.

ДЕ ВТРАЧАЮТЬ

Передача лише набору екранів змушує кожного виконавця самостійно заповнювати прогалини. Це створює різні версії одного сценарію та повертає команду до повторних погоджень.

ЧЕКЛІСТ РОЗДІЛУ

- [ ] Передайте назви екранів, їх призначення та зв’язки між ними.
  Без карти переходів виконавці бачать екрани ізольовано й не розуміють послідовність сценарію.
- [ ] Додайте до кожної критичної кнопки, форми та навігаційного елемента конкретний коментар.
  Без контексту одна й та сама область може отримати різну поведінку в дизайні та розробці.
- [ ] Позначте змінний через CMS контент і правила його відображення.
  Без цього редакторські можливості або технічні вимоги доведеться уточнювати після старту реалізації.
- [ ] Переконайтеся, що дизайн і розробка працюють з однією погодженою версією структури.
  Без єдиної версії зміни з різних файлів і коментарів швидко розходяться.

ЧАСТІ ПИТАННЯ

Чим варфрейм відрізняється від прототипу?

Варфрейм фіксує склад екрана, порядок елементів, навігацію та логіку дій. Прототип дає можливість пройти переходи між екранами й перевірити поведінку сценарію після взаємодії.

Чи потрібен варфрейм для лендінгу?

Так, якщо команда по-різному розуміє структуру, форму або цільову дію сторінки. Якщо сценарій очевидний, а зміни локальні, може вистачити швидкого ескізу або точного коментаря до макета.

Які стани обов’язково показувати у складному сценарії?

Потрібно показати звичайний екран, порожній стан за відсутності даних, помилку заповнення або надсилання, успішне завершення дії та спосіб повернення до сценарію. Для різних ролей також фіксують відмінності в доступному контенті й правах.

Коли варфрейм уже можна передавати дизайнеру та розробці?

Коли названі екрани, показані зв’язки між ними, описані критичні дії, поля, стани та правила доступу, а відкриті питання мають зафіксовані рішення. Передавати потрібно одну погоджену версію з коментарями до конкретних елементів.

Чому не варто одразу переходити до дизайн-макета?

Дизайн-макет визначає вигляд, але не усуває невизначеність щодо логіки сценарію. Якщо структура, переходи або наслідки дій не погоджені, помилки проявляться пізніше, коли їх складніше обговорювати та виправляти.

ЩО ДАЛІ

Обговорити з Apricode

apri-code.com/kontakty/