Чек-лист
для разработки с ИИ

Вопросы, которые нужно обсудить с ИИ перед началом разработки, чтобы избежать ошибок и получить рабочий код

FRONTEND

02

Архитектура и стек

Архитектура

- → Какой паттерн: `SPA`, `MPA` или микрофронтенды?
- → Какая структура папок и модулей?
- → Как управлять состоянием: `Redux`, `Zustand`, `Context`?
- → Какой роутинг и как обрабатывать навигацию?

Технологический стек

- → Фреймворк: `React`, `Vue`, `Next.js`?
- → Сборщик: `Vite` или `Webpack`?
- → Стилизация: `Tailwind`, `CSS Modules`, `Styled`?
- → Какие библиотеки для UI-компонентов?
- → TypeScript или JavaScript?

FRONTEND

03

Безопасность и UX

Безопасность и производительность

- → Как защититься от `XSS` и `CSRF`?
- → Нужен ли `Code Splitting` и `Lazy Loading`?
- → Как кешировать данные на клиенте?
- → Оптимизация изображений: `WebP`, lazy load?
- → Как обрабатывать ошибки на клиенте?

UX/UI стандарты

- → Какие требования к адаптиву и кроссбраузерности?
- → Нужна ли доступность `WCAG`?
- → Как валидировать формы и показывать ошибки?
- → Как обрабатывать состояния загрузки?
- → Интернационализация нужна `i18n`?

BACKEND

04

Концепция и архитектура

Концепция проекта

- → Какая основная цель проекта?
- → Тип системы: `SaaS`, маркетплейс, чат, AI-сервис?
- → Нужна ли авторизация пользователей?
- → Какие роли и права доступа?
- → Какие основные сущности и связи между ними?

Архитектура

- → Монолит или микросервисы? Почему?
- → Какие сервисы нужны: auth, chat, notifications?
- → Способ коммуникации: `REST`, `gRPC`, очереди?
- → Нужен ли `API Gateway`?
- → Как разделить на слои: controllers, services, repositories?

BACKEND

05

Стек и API

Технологический стек

- → Язык: `Python`, `Node.js`, `Go`?
- → БД: `PostgreSQL`, `MongoDB`, `MySQL`?
- → ORM: `Prisma`, `SQLAlchemy`, `TypeORM`?
- → Нужен ли `Redis` для кеша или очередей?
- → Фреймворк: `FastAPI`, `Express`, `NestJS`?

API

- → Какие эндпоинты нужны: auth, users, products?
- → Какие методы и параметры у каждого?
- → Какой формат ответов и коды ошибок?
- → Нужна ли пагинация и фильтрация?
- → Версионирование API: `/v1/`, `/v2/`?

BACKEND

06

Безопасность и инфра

Безопасность

- → Аутентификация: `JWT` или `OAuth2`?
- → Как шифруются пароли: `bcrypt`, `argon2`?
- → Нужен ли `rate limiting`?
- → Защита от `SQL-инъекций` и `XSS`?
- → Как валидировать входящие данные?

Инфраструктура

- → Где хранить `.env` и секреты?
- → Нужен ли `Docker` и `docker-compose`?
- → Как запускать миграции БД?
- → Какое виртуальное окружение?
- → CI/CD: `GitHub Actions`, `GitLab CI`?

BACKEND

07

Масштаб и мониторинг

Масштабирование

- → Как система реагирует на нагрузку?
- → Горизонтальное масштабирование через контейнеры?
- → Как разделяются БД и сервисы?
- → Нужны ли очереди: `Celery`, `Bull`, `RabbitMQ`?
- → Load balancer: `nginx`, `HAProxy`?

Логирование и мониторинг

- → Какие ошибки и события логируем?
- → Система логов: `Sentry`, `Loki`, `Elastic`?
- → Какие метрики производительности отслеживаем?
- → Alerting при критических ошибках?
- → Healthcheck эндпоинты?

ОБЩЕЕ

08

Тесты и документация

Тестирование

- → Какие тесты нужны: `unit`, `integration`, `e2e`?
- → Какой фреймворк: `Jest`, `Pytest`, `Vitest`?
- → Как запускать тесты в CI/CD?
- → Какое минимальное покрытие кода?
- → Моки и фикстуры для внешних сервисов?

Документация

- → Нужен ли `Swagger` / `OpenAPI`?
- → Как генерировать документацию автоматически?
- → README с инструкцией по запуску?
- → Комментарии в коде: `JSDoc`, `docstrings`?
- → Архитектурные решения: `ADR`?