**Front-End архитектура и разработка**

**Замечания и контекст**

Front-End проект должен демонстрировать **осознанный выбор архитектуры**, работу с состоянием, модульность, маршрутизацию, взаимодействие с API и инженерную культуру.

Простой набор страниц без архитектурных решений не считается полноценным FE-приложением.

Важно: **Приложение должно быть SPA** (Single Page Application) и реализовано на современном фреймворке: Angular, React, Vue.js или аналогичном.

**МИНИМАЛЬНЫЕ ТРЕБОВАНИЯ (на 5 баллов)**

**1. Документация**

Обязательные разделы:

- **Архитектурная схема Front-End приложения**
  Модули, компоненты, маршруты, управление состоянием.
- **Описание архитектурного подхода**
  Например: компонентный подход, Flux/Redux-подход, микрофронтенды (если применяются).
- **Документация API**, с которым работает клиент
  Схемы данных, методы, примеры запросов/ответов.
- **User flows / диаграмма навигации**
  Как пользователь перемещается по приложению.
- **Обоснование выбранного стека**
  Почему выбран Angular/React/Vue, менеджер состояния, UI-kit.

**2. Архитектура**

Минимальные архитектурные требования:

- Разделение на:
  - **Компоненты** (UI элементы)
  - **Контейнеры/Pages** (страницы и логические разделы)

    - **Сервисы / Composables / Hooks** (работа с API и бизнес-логикой)
    - **Модули** (для Angular обязательно)
- Использование **роутинга** (минимум 3–4 маршрута)
- Управление состоянием:
    - Локальное состояние +
    - **Глобальное состояние** (Redux/Ngrx/Pinia/Vuex/Zustand/RxJS) или чёткое обоснование, почему оно не требуется.
- Чёткая структура проекта: папки с компонентами, страницами, сервисами, моделями данных, стилями.

Качество взаимодействия с API:

- Работа с реальным API или mock-сервером
- Чёткие интерфейсы/типы данных (TypeScript обязателен для Angular и желателен для React/Vue)
- Централизованная обработка запросов (Axios сервис, Angular HttpClient service, fetch wrapper)
- Обработка ошибок и состояния загрузки

**3. Реализация**

Front-End должен:

- Быть собран через сборщик (Vite/Angular CLI/Create React App/Webpack)
- Иметь компонентный UI
  (не допускается один компонент «всё в одном»)
- Использовать:
    - переиспользуемые компоненты
    - layout-компоненты
    - UI-библиотеку (Material, PrimeNG, Ant Design, Bootstrap) **или собственные компоненты**
- Реализовать:
    - формы (реактивные/контролируемые)

    - валидацию форм
    - таблицы/списки данных
    - фильтрацию/поиск/сортировку (минимум 1 элемент)

Отдельные требования:

- Адаптивная верстка (минимум desktop + mobile)
- Чистый, читаемый код
- ESLint/TSLint + Prettier (или другое форматирование)
- Базовое логирование (например, логирование ошибок в консоль/сервис)

**4. Обработка ошибок**

- Наличие глобального обработчика ошибок (interceptor / error boundary)
- Показывать пользователю понятные сообщения об ошибках (toast/snackbar/dialog)
- Корректная обработка 401/403/404/500
- Skeleton/loading states для всех асинхронных операций

**5. Тестирование**

Требования:

- **Unit-тесты** для минимум 3 ключевых компонентов или сервисов
- **Integration/e2e тесты** (1–2 теста), например:
    - Cypress
    - Playwright
    - Jest + Testing Library
- Тестирование форм, состояний или API-логики

**6. Ограничения**

Что НЕЛЬЗЯ:

- Inline-стили без необходимости, отсутствие архитектуры CSS
- Один компонент на весь проект

- Хранить токены или пароли в коде
- Необработанные ошибки API
- Использование any без обоснования
- «SPA» из статичных HTML страниц без реального UI-фреймворка

**МАКСИМАЛЬНЫЕ ТРЕБОВАНИЯ (на 10 баллов)**

Все минимальные требования **плюс** выполнение 2–3 направлений ниже.

**Направление 1: Продвинутая архитектура**

Если выбрано это направление:

- Микрофронтенды (Module Federation, Web Components, Nx)
- BFF (Backend for Frontend)
- Чёткое разделение на feature-модули
- Lazy Loading модулей и маршрутов
- Clean Architecture / Feature-Sliced Design / Domain-Driven Front-End

**Направление 2: Управление состоянием повышенной сложности**

Если выбрано:

- NGRX/Redux Toolkit с модулями/слайсами
- Reactive state management (RxJS, Signals, Vue reactivity)
- Selectors, эффекты (effects), мемоизация
- Сложные цепочки асинхронных операций

---

**Направление 3: Производительность**

Если выбрано:

- Code splitting и lazy loading
- Мемоизация компонентов
- Использование Web Workers
- Lighthouse ≥ 90 для desktop и mobile

- Уменьшение размера бандла, анализ производительности

**Направление 4: UI/UX расширенной сложности**

- Анимации (Framer Motion, Angular Animations)
- Собственный дизайн-кит компонентов
- Визуализация данных (charts, dashboards)
- Тёмная/светлая тема
- Доступность (Accessibility, WCAG):
    - ARIA атрибуты
    - управление клавиатурой
    - контрастность