САЙТ И КОД В РАМКАХ НЕОПРЕДЕЛЁННОСТИ. КАК СДЕЛАТЬ АДАПТИВНЫЙ САЙТ

Тютюник Н.П., Сергачёв В.Д.,
Ковалева К.А., к.э.н., доцент
ФГБОУ ВО «КубГТУ», г. Краснодар, Россия

Аннотация: В данной статье рассмотрен подход к созданию гибких адаптивных веб приложений. Приведены конкретный примеры и построена архитектура веб приложения.

Ключевые слова: веб разработка, архитектура приложений, uml

Abstract: This article discusses an approach to creating flexible adaptive web applications. Specific examples are given and the architecture of the web application is built.

Keywords: web development, application architecture, uml

В современном цифровом пространстве количество требований к веб-приложениям неуклонно растёт. Сами требования постоянно меняются и старые подходы к разработке сайтов становятся не актуальными. Неадаптивные сайты быстро сталкиваются с проблемами: некорректное отображение на разных устройствах, трудности в интеграции новых данных и функций, а также сложность сопровождения сайта в условиях постоянных изменений.

Поэтому актуальными задачами для веб-разработчиков и системных архитекторов является создание веб-приложений, которые являются легко расширяемыми и гибкими, как с точки зрения пользовательского интерфейса, так и с точки зрения общей архитектуры веб-приложения. В этой статье освещены инструменты и приемы, которые, широко используются на сегодняшний день.

Веб-разработка характеризуется высокой степенью неопределенности, которая может существенно осложнить проектирование и реализацию сайтов. Неопределенность вызвана множеством факторов, которые могут меняться по мере того, как появляются новые технологии и происходят изменения в существующих.

Веб-сайты зачастую полагаются на внешние источники данных, такие как например открытые базы, API открытых сервисов или информация, предоставленная пользователями сети Интернет. Непредсказуемая структура, объем и формат этих данных – одна из главных проблем. Изменение формата данных, выдаваемых API, добавление новых полей в базе данных или внезапное увеличение объема поступающего контента могут нарушить работоспособность сайта.

Разнообразие устройств, браузеров, операционных систем и непредсказуемость действий пользователя создают дополнительную неопределенность.

Один из ключевых принципов проектирования адаптивного сайта – декомпозиция. Этот подход заключается в разделении веб-приложения на независимые, взаимосвязанные модули, каждый отвечающий за конкретную функцию или область контента то есть представляет проект в виде абстрактных уровней каждый от 0 до N требуемых уровней для декомпозиции каждый последующий уровень включён в предыдущий.

Например, сайт новостей можно разделить на следующие модули: Модуль "Страница новостей". Отображает списки новостей с сортировкой, фильтрацией и возможностью поиска. Модуль "Блок категории"**:** Представляет отдельные блоки для разных категорий новостей (политика, культура, спорт). API модуль: Обрабатывает запросы к внешним источникам данных для получения новостных статей. Каждый из этих модулей может быть разработан и развернут независимо друг от друга.

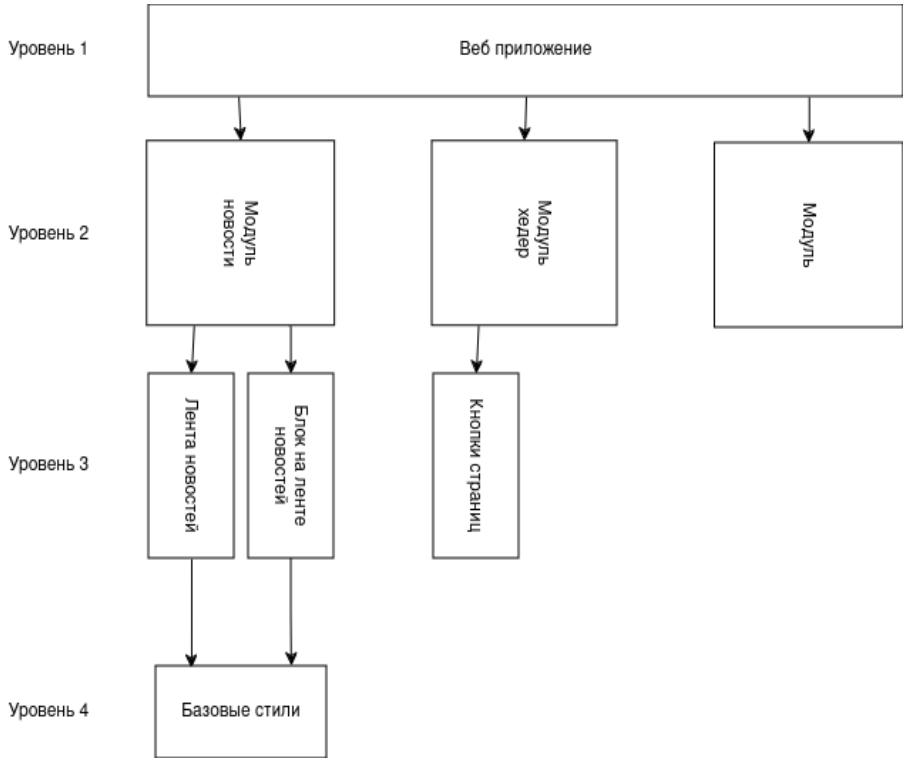

Рисунок 1. Декомпозиция фронтенда

Рассмотрим уровни декомпозиции: 1 уровень это готовое приложение, 2 уровень это страницы сайта, 3 Содержание страницы то есть независимые модули классы функции в случае с React, 4 уровень стили и другие материалы.

В условиях неопределенности особенно важно строить веб-приложения, где модули независимы друг от друга. Это достигается с помощью унифицированных интерфейсов взаимодействия (API), которые служит всенаправленным «языком» для различных частей сайта. Представим для примера сайт как организм из взаимосвязанных органов: каждая часть выполняет свою функцию, но при этом взаимодействует с другими через четко определенные сигналы. API играют роль этих сигналов, обеспечивая обмен информацией между модулями и снижая их зависимость друг от друга. Если какая-то часть сайта меняется или обновляется, ее изменения не обязательно повлияют на другие части, так как они взаимодействуют по стандартизированным интерфейсам. Это повышает гибкость и устойчивость веб-приложения к непредвиденным изменениям данных, требований пользователей или технических ограничений. Теперь рассмотрим фреймворки

для простого поддержания сайта и быстрого изменения содержимого страниц. К ним относятся, например: React, Vue.js, Angular и Next.js. Они позволяют разрабатывать компоненты UI, которые легко обновляются и масштабируются в зависимости от потребностей.

Важную роль в наполнении сайтов играет правильно спланированный backend. Он обеспечивает хранение данных, обработку запросов, логику взаимодействия между модулями и интеграцию с внешними сервисами.

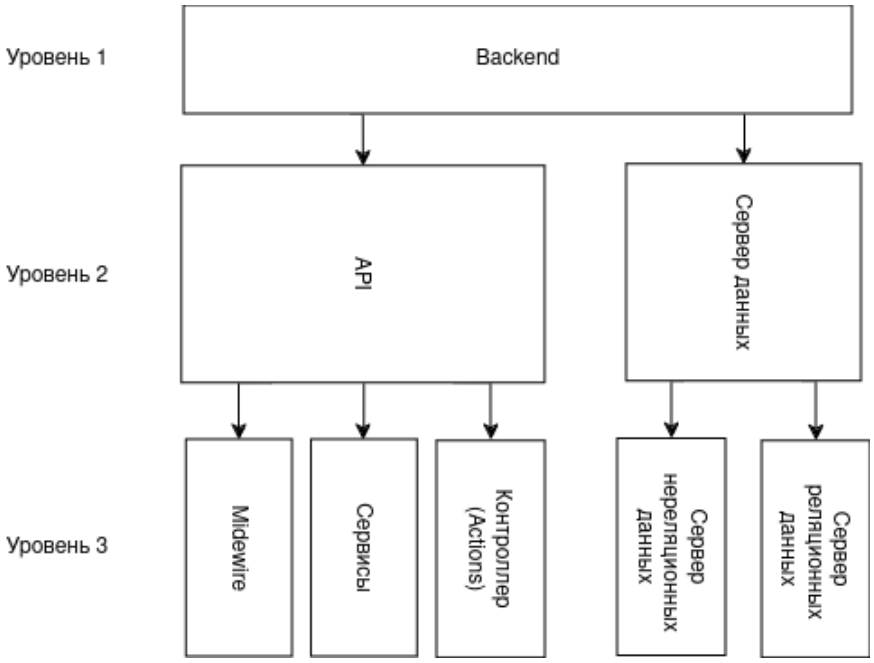

Рисунок 2. Декомпозиция бэкенда

Рассмотрим уровни декомпозиции: 1 уровень - это общее представление бэкенда. На уровне 2 представлены две базовые подсистемы бэкэнда: сервер данных, хранящий всю информацию, которой наполняется сайт, а также API - система, которая обрабатывает запросы, поступающие со стороны фронтенда для наполнения сайта содержимым. 3 уровень описывает промежуточные обработчики (англ. middlewares), сервисы, контроллеры, а также данные, хранимые на сервере в определенном формате. Обработчики среднего уровня осуществляют предварительные анализ и преобразование запроса перед тем как передать его в контроллер. Контроллеры - это основные функциональные составляющие в бэкенде веб-приложения, построенного согласно

архитектурному шаблону MVC. Контроллер - это компонент, настроенный на обработку запросов по определенному маршруту (или поддереву маршрутов). Когда запрос приходит на сервер и проходит через цепочку обработчиков среднего уровня, далее он обрабатывается контроллером. Сервис - это понятие из методологии Dependency Injection (DI). Сервисами здесь называются модули, реализующие какие-либо интерфейсы. Сервис - основной строительный блок приложения ведь сервис - это класс модуля. Два последних компонента на уровне - данные. Данное построение является оптимальным для создания гибких веб приложений сайтов.

Приведем в качестве примера фронтенд сайта студенческого научного общества (СНО). Используем фреймворк React и TypeScript. Построим модульную систему страниц.

На диаграмме представлена структура главной страницы и её подключенных модулей. IndexPage главный класс страницы index, в нем, при конструировании объекта класса, создаются экземпляры классов News, NewsInfoPage, Header, Footer и отрисовываются в окне браузера.

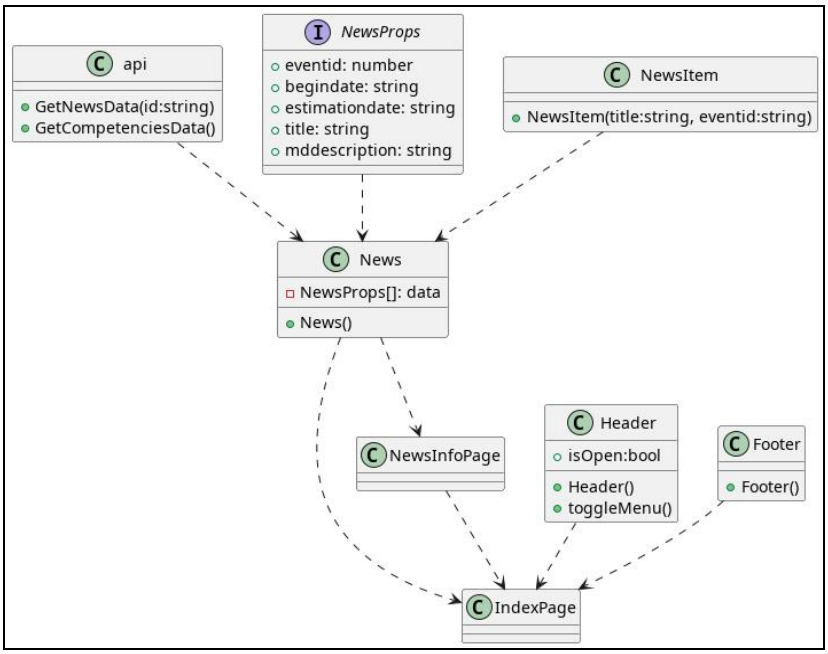

Рисунок 3. Диаграмма классов

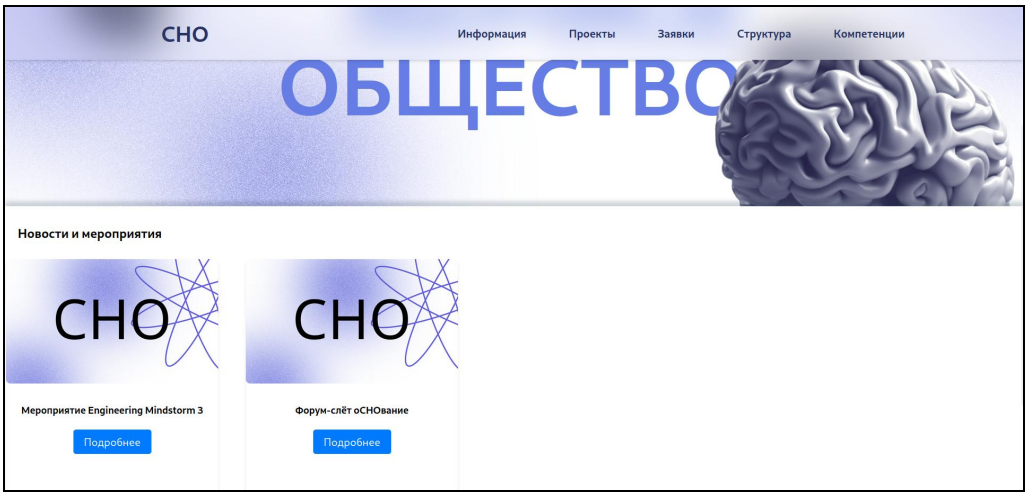

Рисунок 4: Главная страница

Рассмотрим структуру бэкенда студенческого научного общества (СНО).

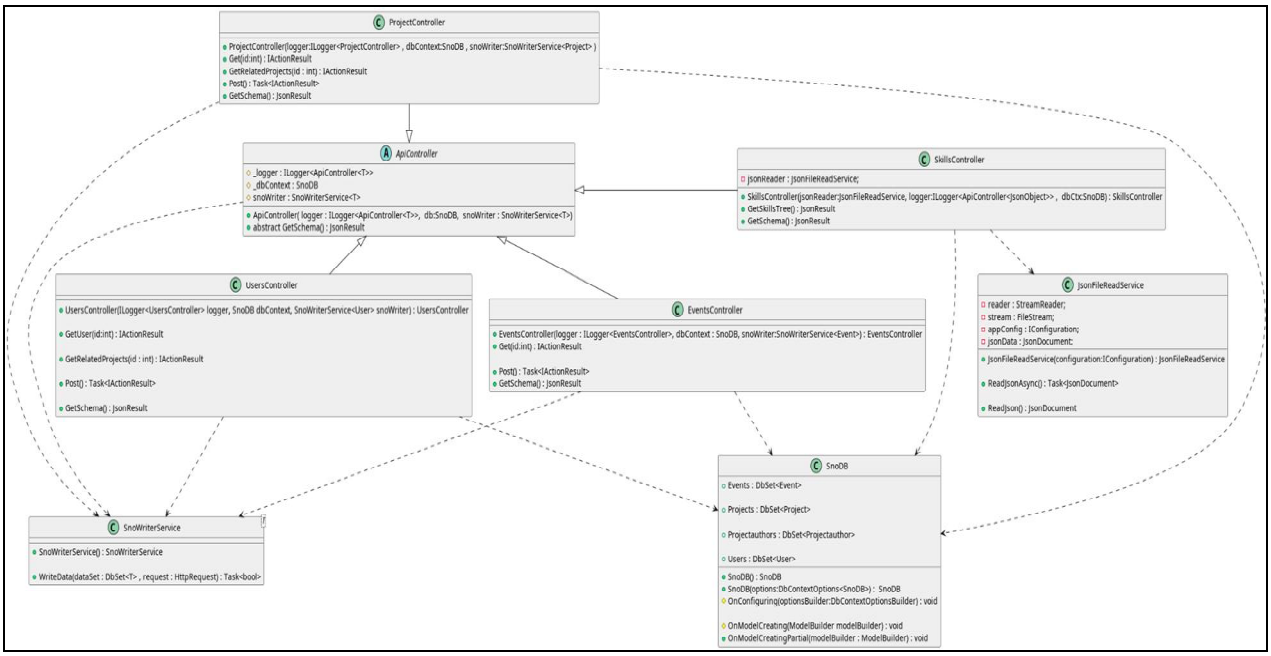

Рисунок 5: Диаграмма классов бэкенда

На диаграмме изображена структура классов бэкенда. На ней можно увидеть иерархию классов-контроллеров. Корнем иерархии является класс ApiController. Он задаёт общий интерфейс и некоторый набор атрибутов и поведений, общих для всех классов-контроллеров приложения. К общим поведениям относятся конструктор и метод GetSchema: по запросу он возвращает клиенту JSON-схему, то есть специальный JSON-объект, несущий в

себе информацию о структуре JSON-объекта определённого типа. От класса ApiController унаследован ряд классов-контроллеров, каждый из которых отвечает за обработку данных своего типа. Наследование – средство достижения повторного использования кода (анг. code-reusability). Благодаря этому сокращается время разработки новых модулей.

В данной статье были рассмотрены архитектурные принципы, инструменты и подробные примеры реализации гибкой архитектуры и для примера реализован сайт. Декомпозиция, модульность, использование унифицированных интерфейсов, фреймворки – ключевые элементы, позволяющие создавать гибкие системы, способные адаптироваться к изменениям.

Литература

1. Аванесян, Д. Н. Использование компьютерных технологий в научной деятельности / Д. Н. Аванесян, К. А. Ковалева // Информационное общество: современное состояние и перспективы развития : сборник материалов XII международного форума, Краснодар, 15–20 июля 2019 года. – Краснодар: ФГБОУ ВО «Кубанский государственный аграрный университет имени И. Т. Трубилина», 2019. – С. 126-128. – EDN KLFPPR.
2. Ивакина, М. Г. Информационные средства защиты информации / М. Г. Ивакина, К. А. Ковалева // Цифровизация экономики: направления, методы, инструменты : Сборник материалов IV всероссийской научно-практической конференции, Краснодар, 17–21 января 2022 года. – Краснодар: Кубанский государственный аграрный университет имени И.Т. Трубилина, 2022. – С. 110-112. – EDN RTQFPX.
3. Комиссарова, К. А. Основы алгоритмизации и программирования / К. А. Комиссарова, С. С. Коркмазова. Том Часть II. – 2-е издание, переработанное. – Краснодар : Кубанский государственный аграрный университет, 2014. – 58 с. – EDN TAGEYV.

4. Михайленко, К. А. Обзор и анализ развития программного обеспечения / К. А. Михайленко, К. А. Ковалева // Актуальные проблемы науки и образования в условиях современных вызовов : сборник материалов II Международной научно-практической конференции, Москва, 04 июля 2021 года. – Москва: ООО "Институт развития образования и консалтинга", 2021. – С. 52-55. – DOI 10.34755/IROK.2021.23.53.085. – EDN TASBTY.

Попова, Е. В. Информационные системы в экономике: методическое пособие для экономических специальностей / Е. В. Попова, К. А. Комиссарова. Том часть 2. – 2-е издание, переработанное. – Краснодар : Кубанский государственный аграрный университет, 2014. – 46 с. – EDN TAGEGJ.